Evolution in Multi-Enzyme Systems

Richard B. Beeby

Doctor of Philosophy
University of Edinburgh
1986

Evolution in Multi-Enzyme Systems

Abstract

A model of early enzyme evolution is developed, in which it is assumed that the high activity and specificity of contemporary enzymes derives from a single cause: the complementarity of enzyme active site and the transition states of the reactions catalysed. This assumption implies that early catalysts had low catalytic activities and poor discrimination between substrates (that is, were multifunctional). A simple multienzyme model of the early cell is given, in which the enzymes have the proposed properties, and kinetic arguments are put forward to show that such a system will evolve high activity, high specificity enzymes if subject to selection for higher growth rates.

Some *ad hoc* assumptions of this model are replaced by more general abstractions, and a case is made for expressing the final model of the initial cell in a modern programming language, and attempting to simulate the evolutionary behaviour of such systems on a digital computer. General questions concerning the simulation of evolving complex systems are addressed in the development of this case. The development of an Ada™ implementation of the model is described.

The results of a number of simulations of populations of cells conforming to the above model are given. The detailed evolutionary course followed is shown to depend on arbitrary events during that evolution, and in each case the population evolves to a local (rather than global) "adaptive peak". Evolution of monofunctionality in all enzymes turns out, in the simulations, to be generally excluded by the details of one of the model refinements. When modification of this refinement, so that monofunctionality becomes a global adaptive peak, is incorporated into a simulation, the population still becomes trapped on a local peak, with the majority of enzymes retaining a multifunctional nature.

It is argued that simulations of the kind presented may contribute substantially to the development of evolutionary theories.

Contents

Chapter 1

Introduction

Complex Systems

A complex system that works is invariably found to have evolved from a simple system that worked. - 15th Law of Systemantics (Gall, 1977).

The only way to understand a complex system is to study something else instead. (Levins, 1970).

Organisms are complex systems. Indeed, they are the most complex systems in our experience. The fifteenth law of systemantics asserts what we intuitively believe must be the case: organisms have not always been as complex as we now find them. The principle is a universal one; Gall's formulation of it is in the context of the design and construction of complex manmade systems and it is frequently cited in the literature on the development of large software projects (e.g. Booch, 1983; Horning, 1985). The so-called *software crisis* of the '70s led to the realisation that the complexity of systems then (and now) being designed and implemented exceeded the understanding of their creators. The incremental (evolutionary) development of such systems has become a necessity in managing their complexity. Similarly, we believe that natural systems of high complexity have evolved from simpler ones, and it is arguable that the development of our understanding of the natural world may increasingly depend on reconstructions of such evolution.

Biological (and, one suspects, cosmological) evolution is contingent, and this limits the degree to which it can be reconstructed. Indeed, the contingency of evolution is, I believe, its single most important feature. Biologists who decry the adequacy of the modern synthesis seek alternative approaches which will somehow introduce determinate, macroscopic evolutionary laws. Hence the

attraction of thermodynamics to evolutionary theorists (Conrad, 1983; Barbieri, 1985; Brooks & Wiley, 1986).

Natural systems, of course, are bound by physical laws which determine their equations of motion (evolution). To understand the evolution of any particular system, we have to know a great deal about it. In engineering, the behaviour of a newly designed system is investigated by constructing a working model of it, a *prototype*. When seeking to understand a natural system we take Levins' advice and construct a model of it (his "something else") for study. The construction of such a model involves a difficult process of abstraction. The 'important' components and interactions of the system must be identified and represented. This selectivity of interest results in the elimination of elements judged to be less essential to the behaviour of the system. In this way, the complexity of the problem is reduced to a more manageable level. This thesis shall be largely concerned with presenting a model of the earliest cellular organisms, expressing it in a modern programming language, and developing and running computer simulations of the evolutionary behaviour of populations of such 'organisms'. Here, the model construction is fraught with potential error, as the systems under study no longer exist to provide a check on the abstraction.

Among the most informative of the models used to understand organisms are those which view the organism as a set of linked enzyme-catalysed reactions. The incorporation of the equations of enzyme kinetics, obtained by the study of isolated enzymes, into descriptions of multi-enzyme systems generates a model which can be used to explain such genetic organismal phenomena as epistasis,

dominance and pleiotropy (Kacser, 1963; Burns, 1971; Kacser and Burns, 1968, 1981). More provincially, within biochemistry, the use of these models to investigate how purely metabolic systems behave and are controlled has generated a wealth of new insights (Kacser and Burns, 1973; 1979; Heinrich and Rapoport, 1974, 1975; Kohn and Chiang, 1981; Kacser, 1983; Fell and Sauro, 1985) and a considerable amount of productive experimental work (e.g. Rapoport, Heinrich and Rapoport, 1976; Flint *et al.*, 1980, 1981; Groen *et al.*, 1982a, 1982b; Tager *et al.*, 1983; Salter, Knowles and Podgson, 1986; Torres *et al.*, 1986; Dean, Dykhuizen and Hartl, 1986; Mazat *et al.*, 1986).

This thesis shall use a multi-enzyme model to address problems related to the early evolution of catalytic proteins. Other areas of evolutionary biology have been illuminated by the judicious use of metabolic control analysis, for example, to investigate population genetical issues concerning the maintenance of variability (e.g. Middleton, 1980; Middleton and Kacser, 1983; Hartl, Dykhuizen and Dean, 1985). An initial outline of the model to be presented and developed here has been published in Kacser and Beeby (1984).

Gould (1980) looks to the emergence of a new and unified theory of evolution. He suggests that one of the characteristics of such a new theory is the reinstatement of the organism at the centre of the biological stage. I believe that the use of a kinetic model of the organism will prove to be very important for the future development of evolutionary theory. The organism can only assume its rightful place if there are appropriate descriptions of it. It is my hope that the present work will illustrate the kind of input that multi-enzyme system theory can make in this area.

The Components of Complex Systems - Enzyme Evolution

Estimates of the number of protein species synthesized by contemporary organisms vary. *E. coli* has probably not many more than the two thousand or so proteins which can be resolved by two dimensional gel electrophoresis (Watson, 1965; Hahn, Pettijohn and Van Ness 1976; Alberts *et al.*, 1983), but the numbers for higher organisms become increasingly uncertain: estimates for mammals increasing from twenty to a hundred thousand or more (e.g. Stern, 1960; Loewy and Siekevitz, 1969; King and Jukes, 1969; Mayr, 1970; Dayhoff, 1978; Clark, 1981). Of course, not all of these proteins are enzymes. Indeed, the number of enzymes known to biochemistry is considerably less, of the order of fifteen hundred (Fersht, 1985). However, many 'non-enzymic' proteins will have a quasi-catalytic role (transport proteins, contractile proteins, etc.). The number of proteins with catalytic effects is, even for the simplest extant cellular systems, in the thousands, and primordial cells are unlikely to have been so well endowed.

This raises a number of questions. What relationship is there between modern proteins and ancient ones? The cell theory tells us that cells arise only from pre-existing cells. Is there an equivalent rule for proteins? If the diversity of protein species has increased, how complex was the ancient metabolic map? If simple, how did it expand? If complex, how is this to reconciled with our assertion that ancient cells had only a small store of genetic information and thus (a valid inference?) only a small number of available catalysts?

It is clear that all extant cells are metabolically complex systems, with a large number of high activity, high specificity, catalysts (enzymes). The origins of these enzymes, and the metabolic transformations they catalyse presents us with the familiar problem of providing plausible routes to account for their evolution. Any account we give in response to this problem will be highly dependent upon our assumptions of the kinetic structure of the earliest cells.

The starting point for this discussion will be the question: if metabolic systems have evolved gradually, then what drives the evolution of enzyme activities? The origin of a new enzyme activity capable of catalysing a step which could contribute to a future metabolic pathway would *prima facie* not be expected to be advantageous until the other enzymes required for the pathway exist. (After all, without such enzymes, the concentration of the substrate for the activity may be zero.) Relying on random, non-selective, events to preserve genetic elements specifying new enzyme activities which in the long term will be useful, as for example in the untranslatable intermediates postulated by Ohno (1970), Koch (1972) and others is unpleasant as a general model of novel enzyme evolution.

An early attempt to circumvent this problem was the *retrograde evolution* theory of Horowitz (1945). Early cells, Horowitz argued, would not have had to synthesize most of their biomolecules: these would have been present in the primaeval 'soup', which would, as its name suggests, have been rich in organic compounds. However, with time, the argument continues, the cells would deplete their environment of essential components and there would be selection to

acquire the ability to synthesize the depleted molecule from any similar compound in the environment. Utilization of such a second compound would then lead to its depletion and selection for acquisition of a catalyst to mediate its production would begin to be felt. In this way, biochemical pathways would be built up, one step at a time; specific catalysts being acquired in reverse order starting with that catalysing the production of the "end product".

Horowitz (1965) later argued, following Lewis (1951), that the second and subsequent enzymes would most probably originate by duplication of the cistron coding for the previous enzyme in the pathway. All the proteins catalysing a pathway would therefore be homologous, and the degree of homology detectable today should increase as one compares enzymes progressively nearer the beginning of the pathway (as these will have arisen most recently).

Models like that of Horowitz, which postulate that relatively specific enzymes are acquired during pathway evolution are described by Chapman and Ragan (1980) as *cumulative* theories. Several advocates of cumulative theories (e.g. Wu, Lin and Tanaka, 1968; Hegeman and Rosenberg, 1970) have rejected the proposal that all enzymes of a pathway are homologous. They argue that the mechanisms of action of different classes of enzyme are too dissimilar to make it likely that it is possible to convert an enzyme of one class into an enzyme of another by only a few amino acid substitutions. It is regarded as more probable that enzyme activities of the appropriate class arise by duplication and mutation of a cistron coding for a pre-existing enzyme of that class (usually acting in some other pathway). This leaves open the question of just how complex the metabolic map of the earliest cells was. What was the origin

of the enzymes catalysing the primaeval metabolic map: are the 'first' instances of each enzyme class non-homologous?. Cumulative theories share the idea of incremental acquisition of enzymes with retrograde direction of pathway growth, as it is this feature which allows a selective theory to be given.

Retrograde evolution theories have been criticised on a number of grounds. It has been pointed out that some intermediates of biochemical pathways are unstable and could not, therefore, have been present in the primordial soup in any significant concentration (Canovas, Ornston and Stanier, 1967; Dagley, 1975); that many intermediates do not readily diffuse through lipid membranes (Hegeman and Rosenberg, 1970); and that, perhaps, the primordial soup was not rich in organic material (Hartmann, 1975).

The widespread use of protein and nucleic acid sequencing techniques have provided some data on whether enzymes of a given class are homologous, and a number of studies have examined enzymes in the same pathway with the question of relatedness among the enzymes very much in mind. Almost invariably, the position is more complex than anticipated. In considering evolution of the enzymes of the glycolytic pathway, for example, Rossmann (1981) concludes that *"there was neither a primordial glycolytic enzyme, nor a series of primordial specific kinases, mutases, etc. Rather, evolution is dependent on the use of domains with simple functions from which are built the enzymes in many metabolic processes."* He leaves open the question of convergence or divergence as the origin of these domains. Yeh *et al.* (1978) found homologous sequences in two enzymes catalysing sequential reactions in the β ketoadipate pathway

from two bacterial genera, *Pseudomonas* and *Acinetobacter*. Subsequent work suggests, however, that the homology derives not from gene duplication, but that some sort of "genetic substitution" has occurred between the genes (Ornston and Yeh, 1979; Yeh and Ornston, 1980). The relationship of alcohol and polyol dehydrogenases between species revealed by Jornvall, Persson and Jeffery (1981) is particularly surprising, with the alcohol dehydrogenase of *Drosophila* being more closely related to the ribitol dehydrogenase of *Klebsiella* than to the alcohol dehydrogenases of mammals, birds, yeast and one of the other bacterial species examined.

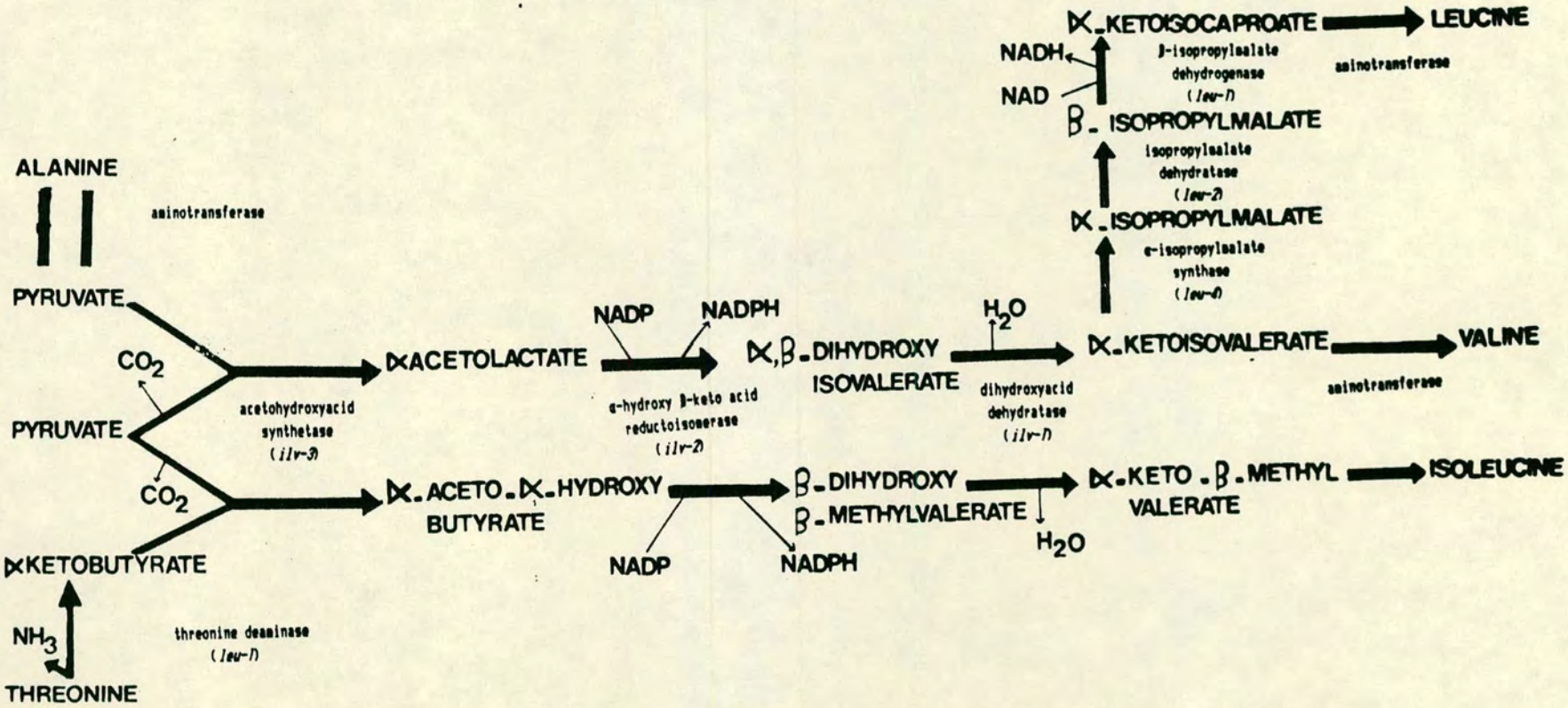
The most important objection to cumulative theories derives from our intuition that modern enzymes are highly improbable structures. On the cumulative view, early cells did not differ qualitatively from modern ones, except perhaps in emergent properties depending on their quantitative differences. Enzymes have always displayed their modern properties of high specificity and reactivity. Early cells were simpler in that they had fewer, not poorer, enzymes than contemporary cells.

Cumulative theories assert that metabolic pathways have grown during evolution as new activities have been created and undoubtedly this is true. Comparative biochemistry reveals that organisms do differ in their biochemical repertoire. But carried to the earliest systems, one finds oneself asserting that in the beginning cells were capable of manufacturing modern enzymes while possessing virtually no synthetic biochemical machinery at all! Furthermore, there are now good grounds to argue that even in modern metabolic evolution 'new' activites are often not created *de novo* but are refined from pre-existing, very

low, activities. These activities occur today in enzymes which are named after their action towards other substrates, but which are *recruited* when novel substrates are encountered. The widespread use of competitive inhibitors in biochemistry illustrates well that enzymes have not evolved so as to exclude all possible competing substrates. Indeed, where extremely high specificity is required, as in the amino-acyl transferases, special proof-reading mechanisms have evolved (Blomberg, Ehrenberg and Kurland, 1980; Fersht, 1981), reinforcing theoretical calculations that there are real physical constraints on specificity. When novel substrates are encountered, then, enzymes may act across pathways, displaying activity *in vivo* towards more than a single substrate. This pathway evolution by recruitment has been described by Jensen (1976). The argument developed here starts with the view that in early metabolism enzymes acting across pathways were the rule, a view expounded by Waley (1969), Ycas (1974), Koshland (1976) and Jensen (1976). The involvement of common enzymes in several pathways does occur in contemporary organisms outside the context of novel substrates. Aminotransferases are typically involved in a number of pathways (Jensen and Calhoun, 1981) and the biosynthetic pathways for isoleucine and valine share almost all of their enzymes, as first revealed by the discovery that auxotrophs for one are also auxotrophs for the other (figure 1.1).

Figure 1.1

The Isoleucine-Valine Biosynthetic Pathways of *Neurospora crassa*



The Improbability of Modern Proteins

The assertion that early enzymes displayed low catalytic rates and low substrate specificity is central to the model to be described. Surprisingly, many consequences of this idea are met with resistance. For example, Barbieri (1985) has attacked Woese's (1967) concept of "statistical proteins" on the grounds that a system with such properties would eventually suffer an "error catastrophe". Yet if early proteins displayed low specificity, the translation machinery of early cells must have been of low fidelity, and produced a stastical ensemble of translation products. Barbieri's own views concerning what might be described as 'statistical ribosomes', which randomly polymerise amino acids rather than translate their sequences from nucleic acids make much higher demands on the informational degeneracy of proteins than Woese does.

The idea that early systems displayed imprecision in molecular recognition appeals to the modern biologists intuition. We *know* that complex biological organisation does not arise *de novo* but evolves. Appeals to intuition, of course, are not authoritative in discussions of this nature. In this section the argument shall be stated in terms of the size of the subset of possible protein sequences that will generate compact stable conformations in the cellular environment.

The action of contemporary enzymes depends upon the *complementarity* of enzyme and substrate (Fischer, 1894; Haldane, 1930; Pauling, 1946), and this in turn depends on a given enzyme adopting a particular, compact, stable conformation. The compactness of the protein confers upon it its stability (Richards, 1977;

Richards and Richmond, 1978), enables the generation of a cavity into which the substrate can bind, and brings into effective orientation chemical groups, often many residues apart on the polypeptide chain, that give rise to a special local environment of high catalytic potential (Lipscomb, 1978).

The ability of a given polypeptide chain to fold into a compact conformation is a highly co-operative property of its amino acid sequence (Privalov, 1963). The co-operativity of folding appears to result in folding intermediates being unstable, both with respect to native and denatured conformations. This means that it is possible to consider the protein, in thermodynamic treatments of folding and stability, as though it can exist in only two possible states (Privalov, 1979).

Anfinsen and Scheraga (1975) believe:

> *Evolution (with thermodynamics dictating the folding) has selected amino acid sequences to form a biologically active molecule, with presumably a limited number of pathways from the unfolded state to a unique native structure of lowest free energy.*

Whether this free energy minimum is global (as Anfinsen and Scheraga, 1975 argue) or local (Levinthal, 1968; Wetlaufer and Ristow, 1973) is of no direct concern here. Indeed, Schulz (1977; Schulz and Schirmer, 1979) asserts that the question cannot be decided. What is important for the current discussion is that for a modern protein to function it must adopt a unique stable globular conformation. For a given arbitrary polypeptide sequence, it has been conventional wisdom that there will be "only a few, if any" (Richards, 1980) compact conformations possible, and that those there are are unlikely to be found by the protein. The number of possible conformations available to a

polypeptide is too large to be explored by the chain during folding (Levinthal, 1968). This is one reason why the folding of a protein must be a cooperative property of its primary sequence. Only a small fraction of possible sequences are likely to possess this property (Flory, 1967, 1969; Edsall, 1968; Lifshits and Grosberg, 1973), and polypeptide sequences must therefore have *evolved* folding pathways which consistently lead to the same final conformational state (see also McLachlan, 1980). Harrison and Durbin (1985) argue that evolution will result in there being a number of alternative pathways leading to the native state, rather than a single path, if it is the case (as in the diffusion-collision model of folding due to Karplus and Weaver, 1976, and the multi-central model of Ptitsyn and Rashin, 1975) that "*native-like local structure dominates the folding process*".

That contemporary proteins have evolved pathways of cooperative folding to a unique compact tertiary structure raises the question of what the folding properties of early proteins were like. If they could not form catalytically active conformations they would presumably have left no descendants. The sequences of early proteins, then, must have been able to form conformations of low stability which, nevertheless, were globular and had catalytic properties. The discovery that compact globular forms can be obtained from random co-polymers of hydrophobic and hydrophilic amino acid residues (Bychkova *et al.*, 1975, 1980; Anufrieva *et al.*, 1975) throws doubt on the conventional wisdom regarding the number of possible compact forms per sequence, as do computer simulations of secondary structure formation in random sequences (Ptitsyn and Finkelstein, 1980) leading to the conclusion that compaction readily occurs. However, it "*cannot be asserted that each such*

globule will have the unique secondary structure" (Ptitsyn and Finkelstein, 1980). The requirement for the evolution of folding pathways leading to a unique final form thus remains, but the problem of what cells did before such pathways evolved is resolved (they generated an ensemble of conformations per sequence, some of which were catalytically active, and selection on these ensembles generated the folding pathways we now observe).

It is thus clear that the requirements of high complementarity of parts of the enzyme surface towards those molecules it interacts with involve much of the amino acid residues of its polypeptide chain(s).

The free energy difference between the unfolded and native states is fairly small for contemporary globular proteins, around 50 ± 21 kJ mol^{-1} (Pace, 1975; Privalov, 1979). Large proteins are composed of a number of compaction units or *domains* (Wetlaufer, 1973; Rossman and Liljas, 1974) which act as cohesive cooperative units with similar stabilization energies (Privalov, 1982).

The relative invariance of this value suggests that it may be an optimum. Much less than a few kilojoules per mole and random thermal motion may severely perturb the protein. While RT is only about 2.5 kJ mol^{-1} at physiological temperatures, energy fluctuations experienced by individual proteins commonly attain 170 kJ mol^{-1}, albeit fleetingly (Schulz and Schirmer, 1979). The margin of safety required above RT must be relatively large, with minimum stabilization energies for compaction units calculated to be 12 kJ mol^{-1}, requiring the involvement of not less than 30 residues (Privalov, 1982). Higher stabilization energies, on the other hand, could be attained only by

larger structural units that would experience considerable folding difficulties (Privalov, 1982). Wetlaufer's (1980) speculative suggestion that such large units, were they to occur, would be degradatively inaccessible during periods of "famine" and have thus been selected against, seems another example of invoking an adaptationist solution to a non-existent problem.

It has been argued, then, that if the negative free energy change stabilizing preferred conformations of primitive enzymes were much less than found in modern proteins, it is likely that an equilibrium ensemble of conformations would have been present for any given sequence. This is especially so if the sequences were short, as many workers have postulated (Cantor and Jukes, 1966; McLachlan, 1972, 1977, 1980; Zuckerkandl, 1975; Von Heijne, Blomberg and Baltscheffsky, 1978). Observations on contemporary small globular proteins suggest that weak secondary forces are insufficient to generate the required stability (Schulz and Schirmer, 1979; Privalov, 1979). In these cases disulphide bridges are found cross-linking the chain, thus reducing the entropy of the unfolded form.

Low stabilization energies in ancient proteins would result in a given chain existing in catalytically active modes only some of the time. Mutations affecting stability would therefore affect the effective concentration of the active forms. If amino acid sequences are thought of as determining the *partition function* of the protein (Blake *et al.*, 1978) then primitive proteins would have had broad partition functions, and mutations confining them around the active conformations would be selectable via their effect on flux through ancient biochemical pathways (see the next chapter on pathway evolution).

Broad partition functions are found in a few contemporary proteins; for example, chromogranin A. Presumably this protein does not require to display complementarity to other cellular components (substrates, co-factors, etc.) for it to function, and may additionally have been selected to avoid crystallizing in the adrenal vesicles where it is found in very high concentrations.

The stability-dependence of complementarity is clearly illustrated by the regulation of activity in some serine proteases. In trypsinogen, and probably chymotrypsinogen, large regions of the protein, including the active site, are disordered (Felhammer, Bode and Huber, 1977). Substrate is not bound, presumably because the binding energy is insufficient to offset the high entropy loss that would be associated with binding (Blow, 1978; Deisenhofer and Huber, 1980). The peptide excision associated with conversion of the zymogen into the active enzyme confers stability on the region, although in chymotrypsin, even at optimum pH, only 85% of the enzyme is in an active conformation (Fersht and Requena, 1971).

The idea that primordial sequences may have been found as an equilibrium of conformations has been expressed by Von Heijne, Blomberg and Baltscheffsky (1978; Von Heijne, Leimar and Blomberg, 1978). They suggest that many "temporary structures" of low stability form during folding and, in small proteins, are stabilized by the formation of crosslinks. The final form adopted by any individual chain depends on which temporary structure it happened to have when the crosslinks formed (different temporary structures favouring the formation of different sets of disulphide bridges). For

primitive proteins this may be a plausible scheme, though it does not hold for contemporary sequences. Bovine pancreatic trypsin inhibitor (BPTI) and ribonuclease, for example, can be denatured simply by reducing their disulphide bridges (e.g. Harrington and Sela, 1959). When these bridges are allowed to reform, protein folding begins. In BPTI the folding pathway depends, for its progress, on the sequential formation of these bonds, and, most significantly, the formation of incorrect bridges (that is, bridges not found in the final native state) are essential in the intermediates if the native form is to be produced (Creighton, 1977a, 1978, 1980). In ribonuclease, a somewhat larger protein, the formation of crosslinks becomes progressively more difficult as folding proceeds. Proteins are eventually produced with three or four disulphide bridges, often incorrect. Thiol catalyzed interconversion of these bonds finally generates the native form (Creighton, 1977b, 1980). Thus, although these bridges are essential for the stability of the native conformation, they do not determine it: that depends on the mutual affinities of regions of the whole sequence. Since in primitive proteins we would not expect the sequences to be as cohesive as they are today, Von Heijne *et al.* (1978) may be right when they argue that equilibrium distributions of forms may have been affected by patterns of disulphide bridge formation. Whether or not this is so, the earlier arguments of low stability and cooperativity in primitive proteins apply and each sequence is likely to have produced a number of different conformations.

That there exists an equilibrium of different conformations for some contemporary enzymes has been postulated by Shnoll and Chetverikova (1975) to

explain fluctuations in the levels of activity observed in a number of systems (see, for example, Duffy, 1971).

Von Heijne *et al.* suggest that differing stabilized conformations of the same primitive sequence may have been functionally different. They may have differed in substrate specificity, or the type of reaction they catalysed. This is a very plausible suggestion. Formally, however, it has no consequences for the model shortly to be presented. Like statistical proteins, the equilibrium ensemble idea underlines the view that a single genetic locus could specify products with activities across a spectrum of the metabolic map of the earliest cells. It will be argued here that broad specificity and multifunctionality were universal properties of the first enzymes, however large or small a part were played by translational and folding variability. They would be characteristics of any particular single conformation. The selection pressures, and responses to them, can be described in the same terms whether or not several active forms were accessible to any given ancient sequence, for selection would see only the function and not the structure.

Finally, it is appropriate to note that the term, *conformation*, implying a static or fixed structure, has some shortcomings when applied to dynamic entities like proteins (see Gurd and Rothgeb, 1979; Karplus and McCammon, 1981). This too, will be passed over as a low-level detail when formulating the model.

Implications of Transition-State Theory for Evolution of Enzyme Catalysis

Transition state theory is a statistical theory of reaction rates. It was formulated in the 1930s by Pelzer and Wigner (1932) and has been considerably developed in the intervening years (Glasstone, Laidler and Eyring, 1941; Laidler and Polanyi, 1965; Laidler, 1969). Despite a somewhat chequered history the theory remains an extremely useful vehicle for studying the kinetics of chemical reactions (see, for example, Mahan, 1974; Pollak and Pechukas, 1978).

The importance of the theory for the current purpose is that it allows the application of the ideas of thermodynamics to be applied to predict reaction rates. Its application to enzyme catalysis (Laidler and Bunting, 1973; Fersht, 1974; 1977) will be reviewed.

The theory assumes that in any reaction pathway there exists an intermediate species of highest energy called a *transition state* which is in equilibrium with the reactants and products of the reaction. For a simple bimolecular reaction, we have the following scheme:

$$A + B \rightleftarrows (AB)^{\ddagger} \rightarrow \text{products} \qquad (1.1)$$

where the transition state is denoted by $(AB)^{\ddagger}$.

In this scheme, an equilibrium constant for the transition state can be formally given:

$$K^{\ddagger} = \frac{[(AB)^{\ddagger}]}{[A][B]} \qquad (1.2)$$

and used in thermodynamic treatments of the reaction. The change in free energy per molecule in moving from the reactants to the transition state is given by

$$\Delta G^{\ddagger} = -kT \ln K^{\ddagger} \qquad (1,3)$$

where k is Boltzmann's constant and T is the absolute temperature.

The theory relates these constants to the rate constant of the reaction, k_1, thus:

$$k_1 = \frac{kT}{h} \exp(-\Delta G^{\ddagger}/kT) \qquad (1.4)$$

where h is Planck's constant.

The following is the standard scheme for a unimolecular enzyme-catalysed reaction obeying Michaelis-Menten kinetics:

$$E + S \underset{k_{-1}}{\overset{k_{+1}}{\rightleftarrows}} ES \xrightarrow{k_2} E + P \qquad (1.5)$$

In the mechanism proposed by Michaelis and Menten in 1913 it is assumed that the enzyme-substrate complex, ES, is in equilibrium with free enzyme and substrate (that is, that $k_2 \ll k_{-1}$). There is a dissociation constant for the complex given by

$$\frac{[E][S]}{[ES]} = K_s \qquad (1.6)$$

and the total concentration of enzyme, $[E]_o$, is the sum of $[E]$ and $[ES]$. Given these assumptions, and that $[P]$ is small, the rate of the reaction, v, is

$$v = \frac{[E]_o[S]k_2}{K_s + [S]} \qquad (1.7)$$

This has the same form as the familiar Michaelis-Menten equation of saturation kinetics, which does not depend upon the assumptions made above:

$$v = \frac{[E]_o[S]k_{cat}}{K_M + [S]} \qquad (1.7a)$$

where $K_S = K_M$ and $k_2 = k_{cat}$. In other mechanisms the physical interpretation of the Michaelis and catalytic constants differs. In the context of using transition state theory note that when $[S] \ll K_M$ one can treat the collective, k_{cat}/K_M, as an apparent second-order rate constant:

$$v = \frac{k_{cat}}{K_M}\ [E]_o[S] \qquad (1.7b)$$

so that

$$k_{cat}/K_M = \frac{kT}{h} \exp(-\Delta G^{\ddagger}/kT) \qquad (1.8)$$

In the course of the current work there shall be a great deal to say about k_{cat}/K_M. It is a measure of the relative reactivity of an enzyme towards competing substrates, and on the assumption, developed later, that evolution maximises enzyme activity, it is this collective, which Fersht (1977) calls the *specificity constant*, that will be our main concern. A "perfectly evolved" enzyme (one that is diffusion limited) has a value for the specificity constant of around 10^8 $s^{-1}M^{-1}$, and $K_M > [S]$ (Fersht, 1977; Brocklehurst, 1977).

The value of K_M being high means that the enzyme binds its substrate weakly. Haldane (1930) suggested that the binding of the substrate to the enzyme could be used to distort the substrate towards the product. This idea was taken up by Pauling (1946) who proposed that an enzyme should be complementary to the transition state rather than the substrate. The contrasting demands of low binding of substrate and high binding of transition state clearly impose constraints on the catalytic potential of enzymes, depending on how similar these two species are. Evolving an enzyme so that it is complementary to the

transition state allows the increase in binding energy in forming the transition state $ES^{\ddagger}$ to be utilized in catalysis (in transition state theory terms, in lowering the activation energy of k_{cat}/K_M) (Jencks, 1975; Fersht, 1977). The assumption that enzymes evolve to be complementary to the transition state has been useful in studying the mechanisms of particular enzymes (see, for example, Kraut, 1977), and has recently had direct experimental support (Leatherbarrow, Fersht and Winter, 1985).

The point to be emphasised in the current discussion is that the response of the Michaelis and catalytic constants to evolutionary pressures for higher catalytic rates cannot be independent of each other. High specificity (obtained in an enzyme which has maximal complementarity to a given transition state) is not an "extra" of enzymic catalysis, it is central to it. In the present work, it is argued, following Waley (1969), Ycas (1974), Koshland (1976) and Jensen (1976), that early enzymes had low specificities and low turnover numbers (high K_M, low k_{cat}, low k_{cat}/K_M for all substrates). The above analysis suggests the route that enzyme evolution would take if selection were for higher rates of catalysis. The model to be presented here points out a problem for such evolution, and a solution to that problem.

Multi-Enzyme Systems

The stability, catalytic activity and specificity of primordial catalysts, it has been argued in the earlier parts of this Introduction, were low. The model of catalytic evolution to be described is founded on these assumptions. The framework into which these ideas are to be fitted is that the systems of which these *proto-enzymes* were components can be described as a metabolic map, the individual steps of which are kinetically realised by the proto-enzymes. That is, the primordial cell can be abstractly treated as a multi-enzyme system. Before going on to expound the model in some detail in the next chapter, a brief review of the treatment of multi-enzyme systems will be presented.

Michaelis-Menten kinetics for an enzyme catalysing a unimolecular reaction is described by equation 1.7b. The equation, as any introductory text explains, applies during (initial) reaction conditions where there is no product present. *In situ*, enzymes are kinetically linked by their substrates and products: one enzyme's substrate is another enzyme's product. It is not, therefore, possible to ignore the product of an enzyme in describing its rate. The rate expression for an enzyme catalysing a unimolecular reaction, in the absence of inhibition and allostery, is as follows (Cleland, 1963):

$$v = \frac{k_{cat}}{K_M} [E]_o([A] - \frac{[B]}{K_{eq}})/(1 + \frac{[A]}{K_M} + \frac{[B]}{K_B}) \qquad (1.9)$$

where A and B are the substrate and product of the enzyme with equilibrium constant, K_{eq}, and the Michaelis constant for the reverse reaction is K_B.

If one constructs a metabolic map involving only unimolecular reactions (in doing so one has already performed a considerable simplification of reality) and writes out the rate expression for each enzyme one obtains a set of simultaneous non-linear equations with no explicit solution. An explicit solution can be obtained for certain unimolecular systems, however, if the non-linearity for each step is removed (that is, if $[A]$ and $[B]$ are constrained so that they are small relative to their respective K_M's). In this case the terms involving $[A]$ and $[B]$ in the denominator of the equation become very small and the rate is approximated by:

$$v = \frac{k_{cat}}{K_M}[E]_o([A] - \frac{[B]}{K_{eq}}) \qquad (1.9a)$$

The simplest linear system of unimolecular reactions is an unbranched chain. Such a system, with n enzymes and $n - 1$ variable substrates is illustrated below. The system converts an external metabolite, X_o, into another, X_n (both assumed to have effectively constant concentrations by virtue of some external mechanism or because their *quantities* are very large)

$$\begin{array}{ccccccccc} & & E_1 & & E_2 & & & & E_n \\ X_o & \rightleftarrows & S_1 & \rightleftarrows & S_2 & \rightleftarrows \ \ldots \ \rightleftarrows & S_{n-1} & \rightleftarrows & X_n \end{array}$$

The above pathway will settle to a steady state for any initial set of parameter values. The flux through the pathway, J, assuming a fixed volume, will be identical to the rate of each of the enzymes, $v_1..v_n$. For the sake of brevity, I shall write M_i to represent the K_M of the ith enzyme for the forward reaction, V_i in the place of $(k_{cat}[E]_o)_i$, *and* K_i for the K_{eq} for the reaction $S_{i-1} \rightleftarrows S_i$

$$J = ([X_o]-[X_n]/K_1K_2\ldots K_n) \ / \ (\frac{M_1}{V_1} + \frac{M_2}{V_2K_2} + \frac{M_3}{V_3K_1K_2} + \ldots) \qquad (1.10)$$

The above is an explicit solution for the flux in such a linear unbranched chain of transformations.

The parameters which determine the flux and the other variables of the system (in particular, the concentration of each of the substrates, $S_1..S_n$) are those which occur on the right hand side of the above equation: the (fixed) concentrations of X_o and X_n, the equilibrium constants for each of the reactions in the chain, and the concentrations and catalytic properties (genetically specified) of the enzymes. In creating models of multi-enzyme systems (of which this is the simplest) it is possible to answer questions about the effects of varying the values of the parameters of the system, on its variables.

In the model of early metabolic systems to be introduced in chapter 2, the effects of altering the enzyme parameters (in particular, $[E_i]$ and the respective specificity constants $(k_{cat}/K_M)_i$ for each of the enzyme's substrates) on the dependent flux will be the principal question of concern.

A useful measure of the effect of altering the activity of an enzyme (its *effective concentration*) on a dependent flux is the *Control Coefficient* (Higgins, 1965; Burns, 1971; Kacser & Burns, 1968, 1973; Heinrich & Rapoport, 1974). This is the ratio of the fractional change in the flux to the fractional change in effective enzyme concentration which caused it, expressed in the limit as a differential:

$$C_i = \frac{dJ}{J} \Big/ \frac{d[E_i]}{[E_i]} \qquad (1.11)$$

or, for a small, finite change, δ, in effective enzyme concentration

$$C_i \simeq \frac{\delta J}{J} \Big/ \frac{\delta [E_i]}{[E_i]}$$

There is an explicit analytic solution for the values of each of the control coefficients for a linear unbranched chain of unimolecular reactions (Kacser & Burns, 1973).

There is a fundamental theorem at the heart of the modern theory of control of metabolic pathways: the sum of all the control coefficients affecting a flux is always unity (this is called the *Summation theorem*), a result that is believed for any multi-enzyme system, however complex and non-linear it may be (Kacser & Burns, 1973). The summation theorem is easy to prove for the linear system described here (one simply adds all the explicit expressions for the coefficients) but the general proof given by Kacser & Burns depends on the demonstration that increasing the concentration of all enzymes in a system by the same factor increases the flux through the system by that same factor. Experimental verification for particular natural (i.e. complex, non-linear) systems (for example, Groen *et al.*, 1982a; Salter *et al.*, 1986) exists. It seems intuitively obvious that the issue of control coefficients is concerned with the distribution of control of a metabolic system among its components, and as with any question of distribution, it is only possible to hand out what there is. The sum of all the parts must equal the whole. One important consequence of the theorem, which has ample observational support, is that the statistical expectation of the value of the flux control coefficient for any given enzyme is small, accounting for the almost universal recessivity of null enzyme mutations (Kacser & Burns, 1981).

Although the theoretical treatment of multi-enzyme systems is far advanced, dealing with systems of considerably greater structural and kinetic complexity than the linear case briefly described (Burns, 1971; Kacser & Burns, 1973, 1979; Heinrich & Rapoport, 1974, 1975; Savageau, 1976; Heinrich, Rapoport & Rapoport, 1977; Kacser, 1983; Fell & Sauro, 1985), linear systems will be at the heart of the work to be described. The reasons for the simplification will be discussed later, but depend on the computational costs involved in the treatment of non-linear systems. The extent to which the generality of the model is affected by this reduction in the complexity of the system, will be a matter for considerable discussion. The generality of the summation theorem demonstrates that important features of non-linear systems can be captured in simple linear models and provides some hope that the convenience of explicit, analytic solutions has not exacted too high a price.

Chapter 2

A Model of Catalytic Evolution

The Kinetic Structure of Early Cells

It is a central part of the problem addressed here that there is a sense in which (the components of) modern organisms are extremely improbable structures. The Introduction was concerned with the improbability of modern proteins, and repeated the view that folding into a unique compact three-dimensional globule is a cooperative, sequence-dependent, property of proteins which a randomly generated polypeptide would be unlikely to possess. Discussions on the origins of metabolic systems frequently centre on this improbability and assert that the chances of a randomly generated sequence possessing a certain catalytic activity (say, pyruvate kinase) is immensely small.

Of course, it is possible to state how many possible polypeptide sequences there are for the set of polypeptides of length N comprised of M different kinds of amino acid residues: namely M^N. When, as in modern proteins, the value of M is twenty, and of N anything from fifty to fifteen-hundred, any particular sequence is extremely improbable indeed. Quastler (1964), for example, considered that for prebiotic processes to have generated functionally meaningful sequences by "random" polymerisation, the fraction of residues in which meaning resides must be very small (perhaps less than ten amino acids in a typical protein). As this is clearly not the case some have concluded (see, for example, Yockey (1977), Hoyle (1985), and fundamentalist writings such as Wysong (1976)) that such processes did not happen, and Darwinian evolution in proteins does not occur. Even without drawing such conclusions,

much concern over the improbability of meaningful macromolecular sequences has been expressed (Salisbury, 1969 is a good example).

John Maynard Smith (1961, 1970) introduced the idea of *protein space* in a lucid and disarming discussion of this problem. Consider the set of all protein sequences of length N, and represent each sequence as a point in an N-dimensional space. Similar sequences are close together in space (think of it as a $20 \times 20 \times \ldots \times 20$ matrix like the $4 \times 4 \times 4$ matrix sometimes used to represent the genetic code), so that a single amino acid substitution will transform a sequence into a directly adjacent one.

For evolution by natural selection to be possible, meaningful sequences must form a continuous network in protein space. There must, on the average, be one or more amino acid substitutions for a given functional sequence which result in an equivalent or superior sequence. (This does not deny the existence of peaks of one or a few sequences which are better than all their neighbours.) If this condition is met, then the requirement that random polymerisation processes "hit upon" stable, high-activity polypeptide sequences is much relaxed. It is only necessary that such a sequence be accessible by a series of single amino acid substitutions which maintain or improve function from a sequence with low effective activity. This leads to the following critical assertion: the fraction of polypeptide sequences which have some functional activity is very large (though the level of that activity will usually be low).

Koch (1972), doubts the network hypothesis, though no reasons are provided. For Koch, single substitutions could not have been the route to modern

activities. Instead, the evolution of high activity enzymes from low activity ancestors would frequently have required to make several simultaneous changes in a polypeptide sequence, where each individual change would impair or eliminate activity. The chances of such changes actually occuring simultaneously are vanishingly small, and so Koch looks to the accumulation of mutations in untranslated gene copies (originally duplicated, perhaps, to increase activity: see below) to provide a bridge across the gaps in the network. The approach of the current work disagrees with the assumption that makes Koch's hypothesis necessary. It is accepted here that the network description is correct, and that the basis of adaptation is precisely this continuity of function in the presence of small changes.

Mutational and interspecies comparisons clearly indicate that there is fairly considerable informational degeneracy in protein sequences (see, for example, Langridge, 1968). The existence of overlaid genes in viruses has enabled one quantitative estimate of the degree of this degeneracy (Sander and Schulz, 1979). The existence of single DNA sequences which can be read in two different frames to generate two different polypeptide products seems extremely improbable in the light of the requirement that each polypeptide sequence meet its sequence-dependent functional requirements. A quantitative estimate of the degree of degeneracy in the functional information of protein sequences can be made by asking what the probability of forming overlaid genes which successfully code for two functions is with different assumptions about the degree of degeneracy per amino acid residue. Sander and Schulz reach the conclusion that at each residue position, on the average, there are a

minimum of four different possible amino acid selections which are consistent with function.

The earliest cells, it is being argued, possessed poor catalysis. The Introduction argued that specificity and activity are complementary features of enzyme action. The catalysts of early cells did not possess high specificity. They were catalytically "sloppy". The kinetic structure of early cells, therefore, was supported by a number (almost certainly small) of poor catalysts. The chemical transformations which occurred would depend on the molecules in the cell's environment, and the particular catalysts the cell produced. In the current work these early catalysts will be called *proto-enzymes*. Each proto-enzyme would catalyse several or even many reactions, and each molecular species would have been involved in reactions catalysed by several proto-enzymes. The metabolic map of the cell would thus have been quite large, even though the number of proto-enzymes in a given cell was small.

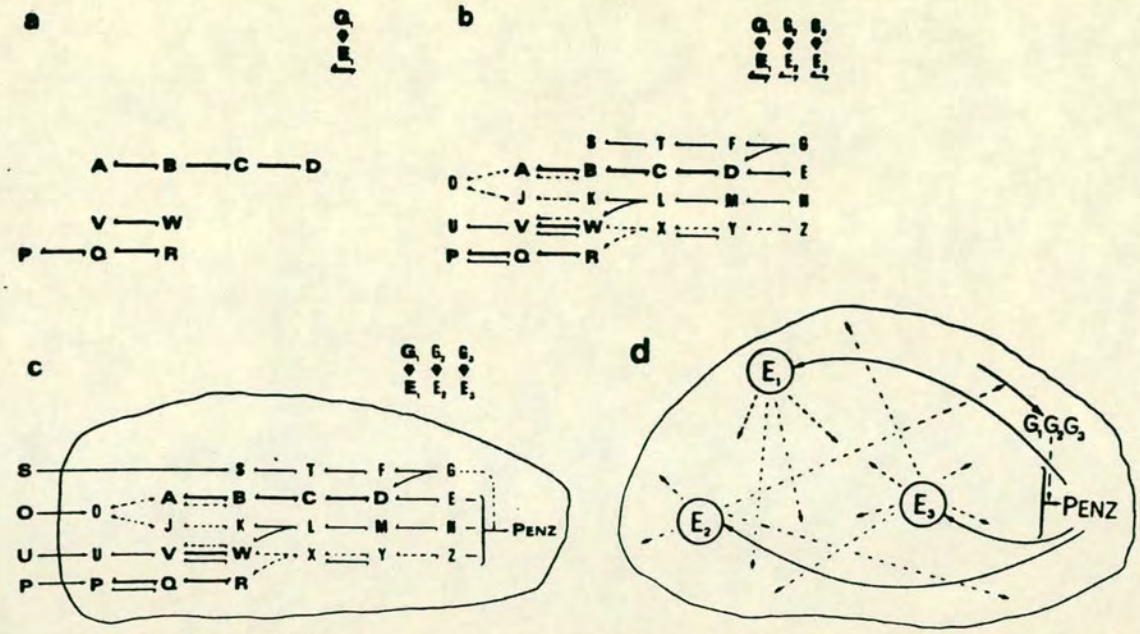
The Early Cell as a Multi-Enzyme System

The kinetic structure outlined above can be expressed in a description of the early cell as a multi-enzyme system. As indicated in the Introduction, the model will deal with linear systems, and initially will restrict its scope to the simplest possible case which incorporates the assumptions described, a metabolic system comprising an unbranched chain of first-order unimolecular reactions.

Conceptually, the model cell is an approximation of a complex system like that shown in figure 2.1. Algebraically, it is treated like the much simpler system depicted in figure 2.2. This complexity-reducing abstraction of the cell will be used in much of the modelling described in later chapters. For the present, it will ease presentation of the formal model, while allowing the broader conceptual model to be dealt with informally whenever possible.

In the Introduction, a description was given of a linear unbranched pathway with n enzymes and $n - 1$ variable substrates. The model for the primitive cell adopted here takes this description as its starting point, though the n enzymatic transformations are now catalysed by less than n *physical* enzyme species.

The end-product of the pathway will be treated as forming the enzymes which catalyse it. This is shown schematically in figure 2.1 as a condensation reaction involving the products of a number of pathways but must remain implicit in the formal treatment given here (to evade its non-linearity).



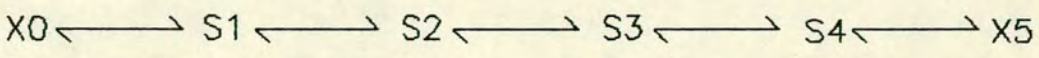
(a) The hypothetical reaction steps shown are catalysed by one enzyme, E_1, which is coded by gene G_1. The reactions are indicated by arrows of equal size, but this does not imply that the catalytic coefficients are equal for all of them. The catalytic effects on all other possible reactions are assumed not to yield reaction rates significantly higher than the spontaneous rate. **(b)** Two more enzymes, E_2 and E_3, with the same type of broad specificity, have been added to the system. The reactions catalysed by the different enzymes are distinguished by the type of arrow. There is some overlap in specificity between the enzymes. **(c)** The complete system is shown, with the supply of external molecular species, S, O, U, and P, and the condensation of products E, N, and Z to form a variety of PENZ, the proto-enzymes, under the influence of G-type molecules indicated. **(d)** A representation of the same system as in (c) showing only the "genes" and enzymes. Not shown, for the sake of clarity, are catalysed steps which do not contribute to PENZ and hence to growth. Such "wasteful" diversion of metabolites would of course occur by the chance properties of proto-enzymes.

From Kacser & Beeby, 1984

Figure 2.1 (above)

Figure 2.2 (below)

Unbranched Metabolic Pathway



The external molecular species, X_0, and the intermediate variable substrates, $S_1..S_{n-1}$, are each recognised and transformed by some proto-enzyme activity. There are, therefore, n proto-enzyme activities present which jointly determine (for any given $[X_0]$ and $[X_n]$) the flux through the system. For each reaction, i, the corresponding proto-enzyme activity is e_i and we can write

$$J = f(e_i) \tag{2.2}$$

where J is the flux per unit volume.

For the moment, there will be no reference in the formulation to the fact that separate activities may reside on the same molecular entity. This will be introduced shortly, but would unnecessarily complicate the presentation at this point.

In a growing system the volume is increasing continuously so that the total flux ($V.f(e_i)$) is also increasing. Throughout the current treatment, the system will be regarded as consisting almost exclusively of the catalytic proto-enzymes. Structural proteins, waste products, and the genetic material will be ignored so that the mass of the system can be considered to be the sum of the (for the moment, n) proto-enzyme species, ΣE_i. The total output in some time, δt, during which the system volume has increased by δV is thus

$$V.f(e_i)\delta t = \delta V.\sum_{i=1}^{n} [E_i] \tag{2.3}$$

(Where the summation is obviously over 1 to n the bounds will not always be explicitly given). For the unbranched chain of linear reactions being

considered here the particular function of e_i giving the flux is, as we have seen in the Introduction, of the form

$$J = \frac{C_x}{\Sigma \ 1/e_i} \qquad (2.4)$$

where the concentrations of the external molecules, X_o and X_n, and their equilibrium constant have incorporated into an environmental constant, C_x, and

$$e_i = (k_{cat}/K_M)_i \ \frac{[E_i]}{K_{Ti}} \qquad (2.5)$$

Clearly the equilibrium constant, K_{Ti} in equation 2.5, is not under the control of the cell (except in the sense that the particular activities present in the cell determine which transformations - and hence equilibria - are relevant). The other parameters on the right hand side of the above equation depend, however, on the genetic specification of the proto-enzyme possessing the activity concerned. The primary structure of the proto-enzyme will determine its tertiary structure, and hence its specificity $(k_{cat}/K_M)_i$, and this structure is specified, however imperfectly, by the appropriate coding sequence (gene). The concentration of the proto-enzyme, $[E_i]$, will depend on its rate of formation, and this, for the purposes of the model, depends only on the concentration of its gene in the cell (which will be expressed shortly in terms of the number of copies of the gene present). It may be that one would expect the existence of structural features that vary from one proto-enzyme species to another that could affect the lifetime of an individual proto-enzyme molecule (how often the covalent links between residues are hydrolyzed) and hence $[E_i]$, but these effects will be disregarded. In terms of the model, then,

the specificity of a proto-enzyme will depend on the structure of its gene, and its concentration on the number of copies of this gene in the cell.

For convenience, the specificity constant for an activity will be condensed into a single *catalytic coefficient*, c_i, with the equilibrium constant for the reaction so that

$$e_i = c_i . [E_i] \tag{2.6}$$

The model, then, will be concerned with genetic variation affecting these two components of catalytic activity.

Growth Rate and Fitness

The rate of growth, G, of the cells represented in the model is given by

$$G = (1/V).dV/dt$$

$$= \frac{J}{\Sigma [E_i]} \qquad (2.7)$$

which, using equation 2.4, is

$$G = \frac{C_\infty}{\sum_{j=1}^{n} \frac{\Sigma [E_i]}{[E_j].c_j}} \qquad (2.8)$$

Any population of cells described by such kinetics would grow exponentially and form a clone. It shall be assumed that there is no sexual reproduction. Mutation rates affecting both the sequences of coding regions and their number would probably have been frequent, for the elaborate error-correcting mechanisms of modern cells would not yet have evolved. The cells would all share the same "life style", there would be no predation. The average relative success of any member of the population, therefore, would depend on its growth rate. As genetic variation results in variation in proto-enzyme composition and activity, it clearly affects growth rate and is selectable.

The fastest growing cells are the fittest.

The Multifunctional 'Trap'

The catalytic effectiveness of the proto-enzymes could only have been altered by changes in their primary sequences. As it has been argued in the previous section that the most important parameter of fitness in the early cells was growth rate, and the growth rate depends on the metabolic activity of the cell, it is natural to look to variation in the structural genes of the proto-enzymes as the proximal targets for selection. Poor catalysts, it seems reasonable to expect, would be have been transformed by mutation and selection into good catalysts.

The problem with this expectation becomes immediately apparent if the kinetic structure of the early cell is interpreted in the light of the observations in the Introduction concerning the complementarity of enzyme and transition state. For the early cell depicted survived by virtue of the imprecision of binding interactions in its proto-enzymes. It was not the case that each activity (e_i) was associated with a different proto-enzyme species. The metabolic map supported by, and supporting, the cell depended on the multifunctionality of the cell's catalysts. It was this multifunctionality which made a large map possible.

The extent to which mutational improvement in a proto-enzyme was possible would clearly be constrained by the requirement to maintain all of its activities (recall that the current scenario ignores wasteful, "unwanted" reactions). Occasionally, transition states for different activities would be very similar to each other (as in the analogous reactions of valine and

isoleucine biosynthesis) and increasing complementarity to one could involve increasing complementarity towards others also, but in general this would not be the case. Even isosteric molecules will differ in properties such as polarity which make substantial binding energy differences, as in the case of valine and threonine.

This argument suggests that the early cell was in a kinetic 'trap'. To increase the activities which sustained it, it had to increase the complementarity of its catalysts to the transition states of the reactions they catalysed, but increasing the activity towards any given transition state could only be achieved by decreasing it towards others: by making the proto-enzymes more discriminatory. This could not be done if it entailed reducing some other (essential) activity associated with the proto-enzyme to zero.

Indeed, it is easy to show that, in general, increasing an activity at the expense of another is likely, for equal fractional changes, to decrease flux (growth rate in the current model) rather than increase it (see figure 2.3). Because the flux depends hyperbolically on any given activity, and the expected value for the activity is on the flat part of the hyperbolic curve, increases in activity will result in considerably less than proportional flux improvements (all other activities being constant, or at least not increasing also). For small decreases in activity, similarly, little is lost (hence the almost universal recessivity of null enzyme mutations, Kacser and Burns, 1981). However, if the loss of activity is large enough to take the activity off the 'plateau' then flux reductions can be become directly proportional to the fractional decrease in activity. This implies that small activity increases,

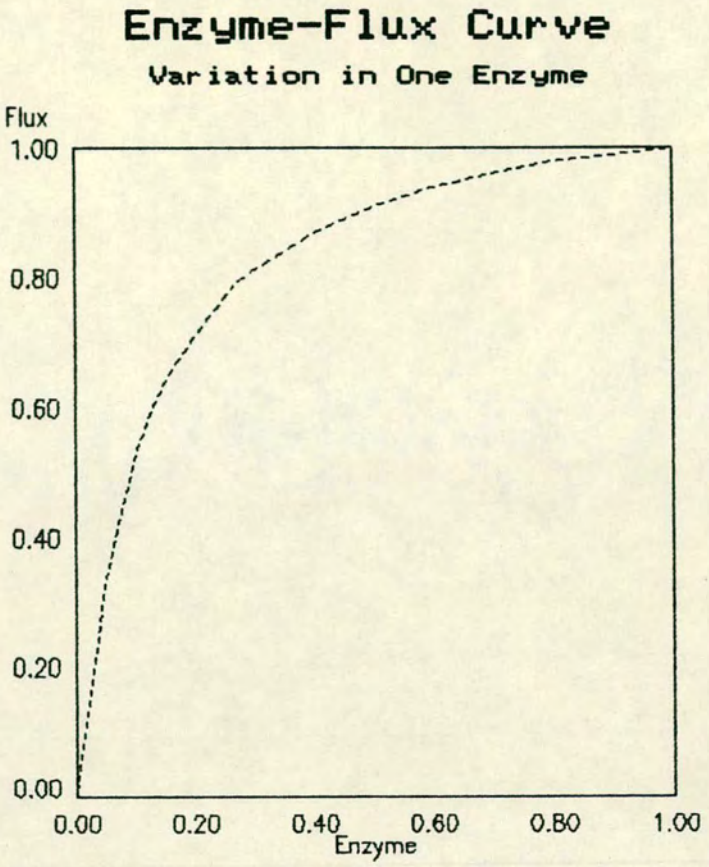


Figure 2.3

The effect of variation a single enzyme on the flux through an unbranched pathway with no saturation. The curve above shows a pathway with ten enzymes, nine of which have identical catalytic coefficients, and the tenth of which is varying in effective concentration. Increasing the length of the pathway (i.e. adding more enzymes) would increase the buffering of the flux.

producing equally small decreases in other activities, may occasionally be growth enhancing, but large changes (and equally, repeated small increases and associated decreases in the same activities) will never be, as long as the mutating gene is solely responsible for the activities in question. In the present scheme increases in a single activity will usually be accompanied by reductions in a number of others, and the possibility of favourable mutation is correspondingly reduced.

The Cost of Allocation

Enzyme concentration occurs in both the top and bottom terms of the denominator of equation (2.8). This expresses the fact that in allocating some fraction of the metabolic output to a given proto-enzyme, i, the fraction available for allocation to other proto-enzymes is decreased. It was stated above that this relative allocation is proportional to the number of genes coding for the proto-enzyme concerned. It is assumed that the early cell possessed no regulatory mechanisms affecting the rates of synthesis of particular proteins (see. for example, Koch, 1972; Jensen and Pierson, 1975).

The question to be addressed here is whether improvements in growth rate, unobtainable by genetic variation affecting c_i, can indeed be attained by genetic changes affecting $[E_i]$, that is, by changes in gene number. Koch (1972) asserts that "duplication serves as an immediate way to augment the amount of any limiting protein species, and no matter how rare duplication events are, an organism bearing multiplicate genes will be selected and become abundant in the population". Koch does not consider multifunctionality, and the term, limiting protein, is left to the readers intuition to define.

Equation (2.8) implies that increasing the concentration of a proto-enzyme species, with concomitant increase in *all* the activities it supports, is not free from cost, because of the decrease in allocation of protein to other proto-enzyme species. This represents an *allocation cost*.

The question of what effect gene duplication has on the growth rate of the early cell can be addressed by reformulating equation (2.8) as follows:

$$G = \frac{C_\infty}{\frac{\Sigma [E_i]}{[E_i].c_i} + \sum_{j \neq i}^{n} \frac{\Sigma [E_i]}{[E_j].c_j}} \qquad (2.9)$$

where the ith activity has been separated from the summation and explicitly shown in the denominator.

The allocation of protein to the ith proto-enzyme of a cell depends on the number of copies of the gene for that enzyme present in the genome of the cell. This number, for the ith proto-enzyme, shall be called n_i, and the proportion of the genome comprising copies of the gene is thus $(n_i/\Sigma n_i)$.

This will not correspond, in general, to the fraction of the total protein in the cell allocated to the ith proto-enzyme, because the proto-enzymes will not have identical molecular weights. Denoting the molecular weight of the ith proto-enzyme by m_i, the relative allocation of protein to that enzyme is

$$\frac{n_i m_i}{\Sigma n_i m_i} = \frac{[E_i]}{\Sigma [E_i]} \qquad (2.10a)$$

To formulate an expression for the growth rate in terms of gene number, so that the effect of gene duplications (or deletions) can be assessed, it is convenient to incorporate molecular weight differences into the catalytic coefficients. First define the molecular weight fraction, α_i, as

$$\alpha_i = \frac{m_i}{\Sigma m_i} \qquad (2.11)$$

so that

$$\frac{[E_i]}{\Sigma [E_i]} = \frac{n_i \alpha_i}{\Sigma n_i} \qquad (2.10b)$$

and then redefine c_i to include α_i

$$c_i = (k_{cat}/K_M)_i \frac{\alpha_i}{K_{1i}} \qquad (2.12)$$

This enables us to express the growth rate in terms of the number of copies of the genes coding for the proto-enzymes by reformulating equation (2.9)

$$G = \frac{1}{\sum_{i=1}^{n} n_i} \times \frac{C_\infty}{\frac{1}{n_i c_i} + \sum_{j \neq i}^{n} \frac{1}{n_j c_j}} \qquad (2.9a)$$

The sensitivity of growth to variation in the number of genes of the *i*th proto-enzyme can be expressed using the *Control Coefficient*, C_{ni}^{G}, which is defined thus

$$C_{ni}^{G} = \frac{dG}{G} / \frac{dn_i}{n_i} \qquad (2.13)$$

(*d* here denoting a partial derivative). This is analogous to the control coefficient referred to in the Introduction, and expressing the local response of the flux to changes in the catalytic activity of the *i*th enzyme. In the current context this flux control coefficient is

$$C_{ei}^{J} = \frac{dJ}{J} / \frac{de_i}{e_i} \qquad (2.14)$$

Reminding ourselves that $G = J/\Sigma[E_i]$, and solving the differential equations above we discover that

$$C_{ni}^{G} = C_{ei}^{J} - n_i \alpha_i / \Sigma n_i \qquad (2.15)$$

This enables us to answer the question of whether changes in gene number represent a way of increasing growth rate. For any particular proto-enzyme, i, increasing the number of genes coding for it will improve growth if the control coefficient it displays towards the flux is greater than its allocation of the total protein. Gene duplication, in other words, is likely to be favourable if the proto-enzyme coded for is catalytically poorer than its partners. As long as $C_{n_i}^{\mathcal{G}}$ remains positive, growth rate increases with n_i. However, as n_i increases $C_{e_i}^{J}$ (and $C_{n_i}^{\mathcal{G}}$) decrease, and eventually a point will be reached where increasing the number of gene copies further will result in the allocation exceeding (or equaling) the (new) flux control coefficient for the proto-enzyme. $C_{n_i}^{\mathcal{G}}$, at this point, becomes negative, and further increases in n_i result in decreasing growth rate.

Intuitively, this is fairly clear. It seems obvious that one cannot go on increasing flux (and growth rate) forever, simply by changing the relative proportions of catalysts. The above reasoning leads to the conclusion that for any given genetic background, there will be an optimum value for n_i (that is, a value giving maximum growth rate).

The Summation Theorem of Kacser and Burns (1979; 1981), states that

$$\sum_{i=1}^{n} C_{e_i}^{J} = 1 \qquad (2.16a)$$

and, by definition

$$\sum_{i=1}^{n} \frac{n_i \alpha_i}{\Sigma n_i} = 1$$

It follows, therefore, that

$$\sum_{i=1}^{n} C_{ni}^{G} = 0 \qquad (2.16b)$$

This means that the distribution of values for the control coefficient of gene number with respect to growth would involve embrace both negative and positive values. Genes with negative values would result in a reduction in growth rate when duplicated. There would also be genes with positive values duplication of which would increase growth rate. Except in the situation where each and every coefficient was zero, there would always be some genes which could be duplicated for increased growth rate, and some which could not be. Multiple copy genes with negative coefficients could give rise to increased growth rate by *deletion* of a copy.

In Kacser and Beeby (1984), on which the above is based, it was stated that, with the exception alluded to, on average half of the genes could be duplicated favourably, and half not (deletions were not treated explicitly). This contains an implicit assumption that the absolute values of the coefficients are uniformly distributed around zero. Whether one grants this assumption or not, the minimum condition, that there be at least one gene which can favourably be duplicated holds (unless all coefficients are zero).

The Optimal Allocation of Protein for Maximum Growth

It follows from the above that, for any given set of genetic sequences, there is a (unique) optimum distribution of protein among their proto-enzyme products for maximal growth rate. Given sufficient time (that is, in the absence of any favourable mutation in the coding sequences, and given appropriate gene duplication and deletion events) the ratio of numbers of copies of these genes in individual cells will come to approximate that required to produce this distribution. The ratios will only approximate the ideal distribution as there cannot be fractional numbers of genes. How close the approximation will be will depend on the particular catalytic values, the route taken and the initial point.

It is possible to state what the ideal distribution of protein is among the proto-enzymes of a cell with the kinetic structure described in the current model. This is the point at which, for each n_i, $C_{n_i}^{\mathcal{G}}$ is zero. From equation 2.15 this is the case when

$$C_{e_i}^{J} = \frac{n_i \alpha_i}{\Sigma\, n_i} = \frac{[E_i]}{\Sigma\, [E_i]} \qquad (2.17a)$$

Referring to the definition of the flux control coefficient (see equation 1.11 and Kacser & Burns, 1973; 1981) and recalling equation 2.12:

$$C_{e_i}^{J} = \frac{1/e_i}{\Sigma\ 1/e_i} = \frac{1/[E_i]c_i}{\Sigma\ 1/[E_i]c_i} \qquad (2.17b)$$

The relative allocation of protein to each proto-enzyme at the optimum distribution can readily be found from the above two equations. Equating the right hand sides of 2.17a and 2.17b, and rearranging, we obtain

$$[E_i]^2 = \frac{1}{c_i} \times \frac{\Sigma\ [E_i]}{\Sigma\ 1/[E_i]c_i} \qquad (2.18)$$

The ratio of the summations on the extreme right of equation 2.18 is the same for all proto-enzymes, so that for each E_i one can write (Burns, 1971)

$$[E_i] \propto (c_i)^{-1/2} \qquad (2.18a)$$

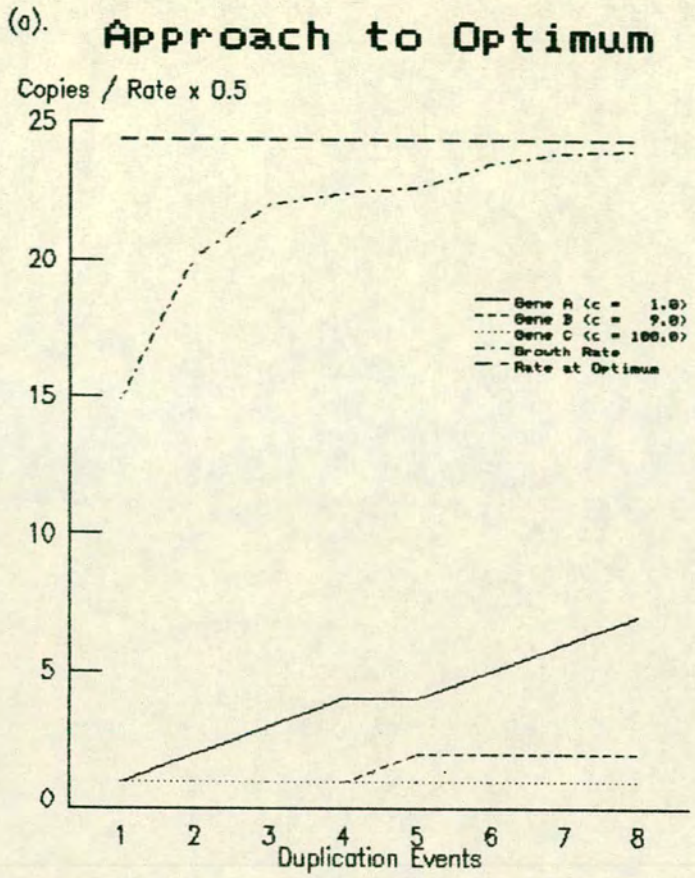
giving the optimum relative allocations

$$[E_1]:[E_2]:[E_3]:\ldots = (c_1)^{-1/2}:(c_2)^{-1/2}:(c_3)^{-1/2}:\ldots \qquad (2.19)$$

The optimal allocation distributes more protein to the catalytically less effective proto-enzymes, to 'compensate' for the poorer catalytic constants.

Some illustrative examples may be instructive. Consider initially figure 2.4a. Beginning with a cell comprising three enzymes with distinct activities, and a single copy of each of the three genes, the graph plots the approach to the optimum allocation against gene number changes where each shown event is that which results in the highest growth rate from the previous point. Only single duplications and deletions are considered. The activities of the enzymes are widely different, so that at the optimum the numbers of the genes for the poorer enzymes are large. At the initial point, duplications of the poorest gene, A, are selectively favoured.

The raw data for the graph are shown in table 2.1, which also gives the total number of favourable (growth rate increasing) duplications and deletions there are from the genotype in any row. In the case given, there is only ever a single favourable duplication (that giving rise to the next row and shown in the graph). Thus, when the number of copies of gene A reaches four a single


(a).
Approach to Optimum
Copies / Rate x 0.5
25
20
15
10
5
0
Gene A (c = 1.0)
Gene B (c = 9.0)
Gene C (c = 100.0)
Growth Rate
Rate at Optimum
1
2
3
4
5
6
7
8
Duplication Events

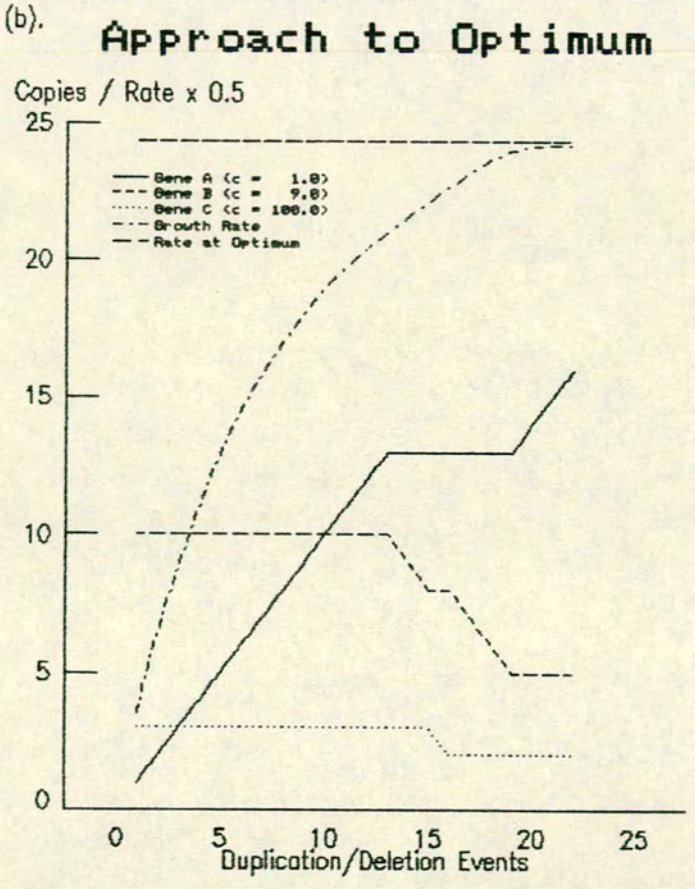

(b).
Approach to Optimum
Copies / Rate x 0.5
25
20
15
10
5
0
Gene A (c = 1.0)
Gene B (c = 9.0)
Gene C (c = 100.0)
Growth Rate
Rate at Optimum
0
5
10
15
20
25
Duplication/Deletion Events

Figure 2.4

TABLE 2.1

NUMBER OF COPIES OF GENE Relative Activities A = 1.0 B = 9.0 C = 100.0			*GROWTH RATE*	*POSSIBLE FAVOURABLE*	
				Duplic.	*Delet.*
1	1	1	29.732	1	0
2	1	1	40.250	1	0
3	1	1	44.010	1	0
4	1	1	44.910	1	0
4	2	1	45.272	1	0
5	2	1	47.071	1	0
6	2	1	47.847	1	0
7	2	1	47.982	0	0
30	10	3	48.675	*Calculated Optimum*	

TABLE 2.2

NUMBER OF COPIES OF GENE Relative Activities A = 1.0 B = 9.0 C = 100.0			*GROWTH RATE*	*POSSIBLE FAVOURABLE*	
				Duplic.	*Delet.*
1	10	3	7.041	1	2
2	10	3	12.959	1	2
3	10	3	17.971	1	2
4	10	3	22.244	1	2
5	10	3	25.907	1	2
6	10	3	29.060	1	2
7	10	3	31.786	1	2
8	10	3	34.149	1	2
9	10	3	36.203	1	2
10	10	3	37.991	1	2
11	10	3	39.549	1	2
12	10	3	40.909	1	2
13	10	3	42.095	1	2
13	9	3	43.196	1	2
13	8	3	44.258	1	2
13	8	2	45.379	1	1
13	7	2	46.479	1	1
13	6	2	47.410	1	2
13	5	2	48.010	1	1
14	5	2	48.271	1	0
15	5	2	48.413	1	0
16	5	2	48.459	0	0
30	10	3	48.675	*Calculated Optimum.*	

duplication of gene B is the only favourable event. Thereafter, only further duplications of gene A are favoured until the final point of the graph/table when all gene duplications result in a reduced growth rate. Note that this point is not the optimum, which cannot be reached from the single-copy condition by a series of single duplications all of which increase growth rate. The ratios at the final point are, however, very close to optimal and the growth rate of the cell is over 98% that at the optimum. Interestingly, in the example given, the optimum is not reached even if the two better genes are at their relative optimum ratios (10:3) at the initial point (figure 2.4b and table 2.2) but in this case there are other duplication and deletion events, not shown, which improve growth rate (though not as significantly as those in figure 2.4b).

When, as in the previous case, enzyme activities differ by orders of magnitude selection can drive the system fairly far towards the optimum even when, unrealistically for most mechanisms of duplication, only single duplications and deletions are allowed. When the activities of the enzymes are less dramatically different, the growth rate improvements will be less and the approach to the optimum more difficult. Consider a three gene system where the relative activities of the "cell's" three enzymes range over a factor of two or three. The example in table 2.3A shows that in such a situation, with one copy of each gene, no single duplication is selectively favoured. The slightly broader range of the example in table 2.3B allows a single duplication of the gene for the enzyme with the lowest activity. The assumption of single copy genes in the initial situation is important in these cases when combined with that of a small genome because the relative cost of a duplication is much

higher than with genomes with larger numbers of genes. As one has every reason to expect that enzyme activities in early systems would have varied over a wide range, as indeed they do today, selection would have been likely to produce populations of cells containing multiple copies of some of their genes.

TABLE 2.3A

GENE *Relative Activities* *A = 16.0 B = 25.0 C = 36.0*			*GROWTH RATE*	*NUMBER OF FAVOURABLE* *Duplic.*	*Delet.*
1	1	1	255.86354	0	0
15	12	10	262.96567	*Calculated Optimum*	

TABLE 2.3B

GENE *Relative Activities* *A = 9.0 B = 16.0 C = 25.0*			*GROWTH RATE*	*NUMBER OF FAVOURABLE* *Duplic.*	*Delet.*
1	1	1	156.04681	1	0
2	1	1	158.17223	0	0
20	15	12	162.96967	*Calculated Optimum*	

The Assumption of Non-Overlap

The above derivation of the optimum allocation of protein among the various proto-enzyme species of the cell involves a major simplying assumption: that the activities (c_i) of each do not overlap. Indeed, the derivation by Burns (1971) referred to above, which treated the case where each enzyme has a single unique catalytic function is (except for minor terminological differences) identical with that given here. The multifunctionality of the species makes no difference.

That each proto-enzyme have a single function is not required in the above analysis, but that each of its functions be unique (not possessed by any other proto-enzyme) is mandatory. Clearly it is not the case in a system containing only monofunctional catalysts that an optimal allocation can involve investing protein in a catalyst performing some essential reaction, x, if there is another catalytic species present in the cell which catalyses x more effectively. In the multifunctional case things are not so simple but the general point that heavy protein investment is made to increase the catalysis of poorer reactions remains. Such investment is only cost-effective when there is no other catalyst one can invest less in for the same effect.

In an earlier presentation of the model (Kacser and Beeby, 1984) this assumption of uniqueness was made implicitly in all the algebra, and for didactic purposes this simplification was undoubtedly justified. In developing the model so that the behaviour of a population can be simulated, however, the point is approaching where it must be discarded and this shall be done in the

next chapter. For the present, the assumption of non-overlap must be borne in mind.

Given the uniqueness of all functions, the irrelevance of multifunctionality to the question of gene duplication and optimal allocation follows from the fact that the expression for the growth rate involves the summation of the catalytic coefficients of the proto-enzymes. The commutativity and associativity of addition allows us to group terms together, for example, thus

$$\Sigma\ 1/[E_i]c_i = (1/[E_1]c_1 + 1/[E_5]c_5) + (1/[E_2]c_2 + 1/[E_3]c_3) + \ldots \qquad (2.20)$$

There is no problem with regarding *each* of the catalytic coefficients as themselves a sum and in thinking of the grouped terms as signifying catalysis by the same physical species (so that, e.g. $E_1 = E_5$). The right hand side of 2.20 can, on these assumptions, be rewritten

$$1/[E_i](1/c_1 + 1/c_5) + 1/[E_j](1/c_2 + 1/c_3) + \ldots$$

(the next chapter will discuss the physical validity of this procedure). On this basis, equation 2.9a can be reformulated as follows

$$G = \frac{1}{n_i + \sum_{j \neq i}^{n} n_j} \times \frac{C_x}{1/n_i(1/c_A + 1/c_B + 1/c_C + \ldots) + \sum_{j \neq i}^{n} 1/n_j c_j} \qquad (2.9b)$$

where the last summation term in the denominator conceals the multifunctionality of the other proto-enzyme species.

Equation 2.9b shall form the basis for the discussion of the evolutionary route taken by cells with the kinetic structure so far described. The solution to the problem posed by the multifunctional 'trap' shall be addressed, retaining the assumption, though somewhat awkwardly once mutation enters the picture, that no catalytic coefficient (say, c_A) associated with some proto-enzyme, E_i, is also associated with E_j.

In the absence of changes to the coding sequences of the genes (mutation) the genetic composition of cells will come to approximate that giving optimal protein allocation to the various proto-enzymes encoded. The existence of translational and folding variability does not affect this conclusion, as long as there is coding. If the structural relationship between gene and proto-enzyme were entirely random, as it would be if the genes of the early cell were non-specific and randomly strung amino-acids together, then evolutionary progress would not be achievable. The quasi-replication of heterosoids in the ribotype theory of Barbieri (1981, 1985) involves such non-evolvable entities and it is difficult to make sense of that theory in the light of such an assumption.

Gene Duplication and the Escape from the Trap

There has been a great deal of speculation in the literature about the importance of gene duplication as a prelude to originating new function (for an extended discussion, see Ohno, 1970). The argument is based on exactly the same kind of premises held here: a new function cannot generally be evolved by altering an existing functional sequence because of the different sequence requirements of such functions. That is, to gain the new function, one must lose the old one. (In the current context, the issue is not the evolution of a new function, but the improvement of an existing one. This shift of emphasis arises from replacing a cumulative model of metabolic evolution with an *en bloc* one.) The traditional argument has looked to fortuitous gene duplication and divergent mutation to 'explain' the evolution of new function. Once a gene has duplicated, then there is an 'extra' copy which can be commandeered for evolving novel functions.

This is unsatisfactory because it relies on chance not only for the duplication event, but also for preserving the duplicated element until such time as it has accumulated enough (individually unselected) mutations to be useful. A major merit of the current model is that it is possible for duplicated elements in themselves to be favoured by selection rather than being simply a 'negligible load'. In this sense, it is not appropriate to regard any of the copies of some gene as being 'surplus to requirements'. Early populations will have been comprised of cells with multiple copies of some of their genes and the question is whether the model can shed light on the expected selective effects of sequence changes in multiple-copy genetic elements.

To make statements about the selective effects of mutations in proto-enzyme coding sequences, one must first be able to state what functional effects such changes may have. Having determined the effect on the catalytic properties of an individual proto-enzyme species, it will be possible to embed the altered catalyst into the system and compare the systems performance (i.e. growth rate) with that of its parent. Hopefully, it will be possible to make some general statements about the directions that the multi-enzyme systems being modelled will take when subject to selection for increasing growth rate.

The Introduction emphasised the importance of complementarity in catalytic function. It is assumed in the current model that early enzymes displayed low complementarity to the molecular species with which they interacted. This is the basis for the argument of multifunctionality in primitive catalysts: a multifunctionality that extended over both substrates and, because of interactions with cofactors, ions and so on, functional classes (see Waley, 1969).

Recall from the Introduction that it is possible to treat the collective, k_{cat}/K_M, (incorporated in the treatment above with their relative molecular weights and the appropriate equilibrium constant into the catalytic coefficients of the proto-enzymes - equation 2.12) as an apparent second-order rate constant so that

$$k_{cat}/K_M = \frac{kT}{h} \exp(-\Delta G^{\ddagger}/RT) \qquad (1.8)$$

(when the conditions given in the Introduction hold).

The argument in the Introduction involved the idea that the enzyme evolves so as to be complementary to the transition state of the reaction. This can best be appreciated by examining $\Delta G^{\ddagger}$, which might be described as the "activation energy of k_{cat}/K_M" (Fersht, 1977).

$\Delta G^{\ddagger}$ can be regarded as having two components: a thermodynamically unfavourable part (that is, involving a positive change in free energy), $\Delta G_A^{\ddagger}$, which is the chemical activation energy required to form the transition state, and a favourable (negative) part, $\Delta G_S^{\ddagger}$, associated with the binding energy of the transition state with the enzyme active site (Fersht, 1977).

$$\Delta G^{\ddagger} = \Delta G_A^{\ddagger} + \Delta G_S^{\ddagger}$$

so that it is possible to rewrite equation 1.8 in the following way

$$k_{cat}/K_M = \frac{kT}{h} \exp(-\Delta G_A^{\ddagger}/RT) \exp(-\Delta G_S^{\ddagger}/RT) \qquad (1.8a)$$

Overall, $\Delta G^{\ddagger}$, is a positive term, but may be lowered by increasing the binding energy, $\Delta G_S^{\ddagger}$, through maximising the complementarity of enzyme and transition state. Such maximisation is brought about when there is a binding site on the enzyme for each of the potential binding groups on the transition state, and these sites are organised so that the binding energy is maximum for each of them: this is what complementarity means. The energetic costs of forming the transition state can be partly paid for by the increase in binding energy associated with its formation (the concept of *utilisation*: Jencks, 1975).

As previously argued, there is a problem here for a multifunctional enzyme which has a single active site interacting with a number of different species. Complementarity to one of these will usually involve exclusion of some or all

of the others. However similar the species are, simultaneous complementarity to all of them is physically unrealizable.

To be able to deal with this formally in the present system, a function to describe the interaction energy of the enzyme-transition state complex is needed. Such a function will require a model of the enzyme active site and the transition state. A very simple model will be presented in the next chapter, but for the present the approach taken by Kacser & Beeby (1984), in which assumptions are made about the relative effects of mutations on transition state binding in the absence of any detailed theoretical view of their nature, will be presented so that the conclusions of that paper can be understood.

The Effects of Mutation on Specificity

The simplest assumption one can make when dealing with the relative effects of alterations in a proto-enzyme active site (caused by mutation in the coding sequence) on the interaction energy of its various substrates (or the transition states they form) is that any change which increases it for one substrate decreases it for all others. Kacser & Beeby (1984), it will be seen, made this assumption, and (for convenience), assumed that the relative change in binding energy was the same for all species.

In accordance with the ideas presented in the Introduction, the relative catalytic constants of a proto-enzyme towards its various substrates is taken to be directly proportional to the binding energy of the enzyme-transition state complex (Fersht, 1977). This is a fairly realistic assumption for similar chemical species interacting with the catalyst in the same way (for example, different species presenting hydroxyl groups for phosphorylation) as the energetic costs of the reaction (the $\Delta G_A^{\ddagger}$ component of the total free energy change) are likely to be very similar (inductive effects, *et cetera*, may differ significantly in particular cases, but can be ignored for simplicity). It is clearly less justified when comparing species upon which different reactions are performed so that, say, a dehydrogenase reaction is being compared with a kinase reaction. That different classes of reaction may be catalysed by the same catalytic species has been considered by Waley (1969) on the basis of the similarity of the cofactors required. At least one case of a single active site catalysing two distinct kinds of reaction has been

suggested (chorismate mutase-phrenate dehydrogenase in *Escherichia coli* and related species: Andrews & Heyde, 1979).

One can now simplify equation 1.8a

$$k_{cat}/K_M \simeq Ce^{\Delta G_S^{\ddagger}/RT}$$

so that the relative specificity constants for two substrates, *i* and *j*, for a given catalyst are given by:

$$\frac{(k_{cat}/K_M)_i}{(k_{cat}/K_M)_j} = \frac{(\Delta G_S^{\ddagger})_i}{(\Delta G_S^{\ddagger})_j} \qquad (2.21)$$

A mutational change increasing the specificity constant for *i* by, say, ten per cent would, on the original Kacser and Beeby model, reduce that for *j* (and any other substrates) by the same proportion. On this assumption, figure 2.5 illustrates the effect on growth rate of increasingly large $\Delta G_S^{\ddagger}$ in a single copy gene. The details of the calculation can be found in Kacser & Beeby (1984). For the particular values chosen a small change in binding energy towards one of the substrates slightly increases growth rate, but progressively larger changes (or repeated small changes) result in considerable rate reductions.

We thus have the following scenario: For any given set of proto-enzymes and substrates comprising a cell, there is an optimum allocation of protein among the catalysts to maximise that cells growth rate. This relative allocation can be achieved by altering the relative numbers of copies of the genetic elements coding for the proto-enzymes. Mutational (sequence altering) changes in a cell which contains a number of proto-enzyme species with multifunctional active

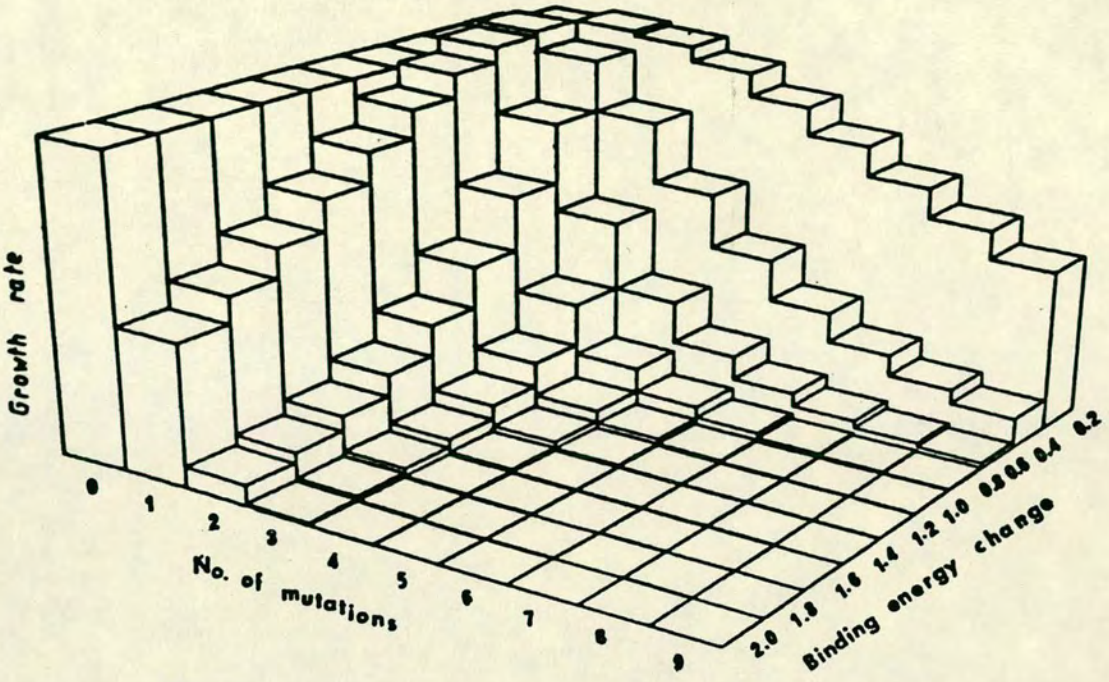


Figure 2.5

The Mutational Trap. The diagram shows the effect on growth of binding energy changes in a multifunctional protoenzyme. The subfunction to be improved has a very low catalytic coefficient compared with those of the other subfunctions - the most favourable case for mutational increase in the growth rate. It is seen that only for small binding energy changes is there a marginal increase in growth rate, which, however, declines after one or two such mutations. After enough mutations any mutation "size" will lead to a decline in the growth rate.

(from Kacser & Beeby, 1984).

sites and non-overlapping activities will be unlikely to be beneficial if there are only single copies of the genes coding for each of the elements, because the multifunctionality imposes high constraints on the active site. However, in any cell population in which individuals have been subject to selection for growth rate there would probably be cells at or near the optimal allocation, and this optimum would almost certainly involve multiple copies of some, if not all, of the genes.

The question posed above was: In the scenario described, what is the fate of sequence altering mutations in multiple copy genes? As the kinetic structure of the cell is explicitly described, this question can be answered by inserting values into the equations of motion of the system and calculating relative growth rates. Retaining the highest value for $\Delta G_s^{\prime}$ used in the previous figure, figure 2.6 shows the effect on growth rate when alterations in affinity are made in one of the copies of a gene represented in the genome with the frequency shown. The activity which is being increased is, as stated above, a poor one, and the gain in the total activity at the step outweighs the loss at other steps, which are buffered by the activities of the proto-enzymes produced by the non-mutated copies of the gene.

Selection, if acting on growth rate as the model assumes, will favour cells with duplications of the catalytically poorest genes. Now we see that mutations in the poorest activities of such multiple copy genes are favoured. As such mutations progress, the activity in question will cease to be the poorest, and mutations in another activity (likely to be on one of the poorest proteins, and hence on a multiple copy gene) will begin to be selected for.

This is the core of the model, that this sequence of duplication and mutation will be selectively repeated until all activities are high and all enzymes are monofunctional. The aim of the present investigation is to attempt to simulate this process, first refining various elements of the model, to determine whether this line of argument is correct in its extrapolation to the modern (monofunctional) condition.

Chapter 3

Refining the Model

One Catalyst - Many Reactions

One problem which has not been alluded to so far, but which may have occurred to the reader, is the problem of competition. If the early proto-enzymes had a single active site which interacted with a number of possible substrates then competition between them for the active site would occur. Each potential substrate would be a competitive inhibitor for all the others.

These competitive effects will give a catalytic advantage, in addition to that already discussed in terms of improved complementarity, to monofunctional enzymes over multifunctional ones, as it is the multifunctionality which creates the competition. Insofar as this is so, the physical factors favouring the evolution of monofunctionality in the presence of selection for higher growth rates are increased.

The problem in the current context is that the competition may call into question the validity of the linear (first-order) approximation used to calculate the fluxes. In this approximation it is not necessary to calculate the intermediate substrate concentrations of all metabolites to determine the flux (equation 2.4) or growth rate (equation 2.8) at steady state. These cancel out of the equations and, as long as the concentrations of the enzymes and external metabolites are known and effectively constant over the time interval of interest, the values of the equilibrium and kinetic constants are sufficient. All this changes when competitive inhibition is introduced. The rate of a single step of a monomolecular enzyme, assuming negligible product concentration, is, in the presence of a competitive inhibitor, I, given by

$$v = \frac{k_{cat}[E][S]}{K_M\left(1 + \frac{[I]}{K_I}\right) + [S]} \qquad (3.1)$$

In effect, each K_M is increased by an amount which depends on its affinity for the inhibitor *and on the inhibitors' concentration.* When the inhibitor is, in fact, another metabolite in the pathway, then the effective K_M of each enzyme must be determined by solving the steady state (to determine the concentrations of all "inhibitors")! To produce such a solution one requires the K_Ms and so must use an iterative method of successive approximation which introduces with a vengeance the kind of computational costs that the linear approach was adopted to avoid.

Fortunately, the model is rescued by the very assumption of first-order kinetics now under discussion. For this assumption is that there is no saturation. The enzymes are operating at substrate concentrations which are low relative to their respective K_M's. The "inhibitors" in question in the model are actually alternative substrates forming part of the chain of reactions the kinetic equations describe. For each proto-enzyme, the "K_I's" are actually the K_M's of its other substrates. Each ratio ($[I]/K_I$ = some $[S]/K_M$) is by hypothesis low, and it was this assumption that was used in the Introduction to derive the linear equation 1.9a from the nonlinear 1.9 in which two $[S]/K_M$ terms, relating to the forward and reverse reactions catalysed by some activity, occur in the denominator. Therefore, the factor by which the K_M must be multiplied to take account of inhibition is approximately 1.

One Reaction - Many Catalysts

The model described in the previous chapter involved a number of simplifying assumptions of varying degrees of importance. One of these, alluded to several times, was that although each proto-enzyme has an active site which reacts with many different substrate species, no two proto-enzymes catalyse the same reaction (the non-overlap assumption). Put in another way, no substrate-product pair react with more than one proto-enzyme.

This assumption is unpleasant firstly because it is implausible. If the early cell's enzymes were indeed 'sloppy' then overlap between their activities would have been inevitable. It is, after all, implicit in the model that a very few catalysts were nevertheless, between them, able to support a viable metabolism. It is highly improbable that the fortuitous properties of these catalysts were such that all (essential) reactions had exactly one catalyst. The assumption is secondly unpleasant because it turns out to be unnecessary (discarding it hardly affects the difficulty of the formal model at all).

Equations 2.18 and 2.19 introduced the notion that the same physical enzyme could be involved in more than one $1/[E_i].c_i$ term without affecting the description of the system in any significant way. It is appropriate at this point to expand using a concrete example, before going on to consider the implications of discarding the non-overlap assumption on the formal description of the metabolism of the early cell.

Consider a cell with a metabolic map comprising an unbranched chain of five metabolic steps which can be described using linear (unsaturated) kinetics.

$$X_1 \rightleftarrows S_1 \rightleftarrows S_2 \rightleftarrows S_3 \rightleftarrows S_4 \rightleftarrows X_2$$

One possible model for the catalytic structure of the cell is that there are five disctinct physical proto-enzymes catalysing the system, so that we have

$$X_1 \overset{E_1}{\rightleftarrows} S_1 \overset{E_2}{\rightleftarrows} S_2 \overset{E_3}{\rightleftarrows} S_3 \overset{E_4}{\rightleftarrows} S_4 \overset{E_5}{\rightleftarrows} X_2$$

and the flux per unit volume through the cell is given by

$$J = C_x \,/\, (1/[E_1].c_1 + 1/[E_2].c_2 + 1/[E_3].c_3 + 1/[E_4].c_4 + 1/[E_5].c_5)$$

This is the simple monofunctional case. In the non-overlapping multifunctional case, one has something like

$$X_1 \overset{E_{1A}}{\rightleftarrows} S_1 \overset{E_{1B}}{\rightleftarrows} S_2 \overset{E_{2C}}{\rightleftarrows} S_3 \overset{E_{2D}}{\rightleftarrows} S_4 \overset{E_{3E}}{\rightleftarrows} X_2$$

where enzymes with the same numeric subscript are the same physical species, and the differing subscript letters refer to different catalytic activities so that the flux is now given by

$$J = \frac{C_x}{\frac{1}{[E_1]}(1/c_{1A} + 1/c_{1B}) + \frac{1}{[E_2]}(1/c_{2C} + 1/c_{2D}) + \frac{1}{[E_3]}(1/c_{3E})}$$

Both the above flux equations are expansions of equation 2.4 (with 2.6 incorporated) involving summation over the five different catalytic activities present

$$J = \frac{C_x}{\sum_{i=1}^{n} 1/[E_i].c_i} \qquad (2.4a)$$

In terms of the current discussion, the important difference between the two cases involves the expression for the growth rate. In the monofunctional case this is

$$G = \frac{C_x}{\sum_{j=1}^{n} \frac{\sum_{i=1}^{n} [E_i]}{[E_j].c_j}} \qquad (2.8)$$

and the number of catalytic activities and physical enzymes is the same (five in the particular case in point). In the multifunctional case, however, there are more activities (five) than there are enzyme species (three). The summation indexed by i is not actually over 1 to n but over 1 to the number of enzyme species (call this s) and so involves less terms than that indexed by j:

$$G = \frac{C_x}{\sum_{j=1}^{n} \frac{\sum_{i=1}^{s} [E_i]}{[E_j].c_j}}, \quad s < n \qquad (3.2)$$

Now consider a third case in which different proto-enzymes may display catalytic activity towards the same reaction step

$$\begin{array}{ccccccccccc} & & E_{1A} & & E_{1B} & & & & E_{1D} & & \\ & & & & E_{2B} & & E_{2C} & & & & E_{2E} \\ & & E_{3A} & & E_{3B} & & & & & & E_{3E} \\ X_1 & \rightleftarrows & S_1 & \rightleftarrows & S_2 & \rightleftarrows & S_3 & \rightleftarrows & S_4 & \rightleftarrows & X_2 \end{array}$$

There are three different proto-enzymes in the example, and there are metabolic steps which are catalysed by one, two or all of these catalysts. This model is a much better approximation of figure 1.1 than the multifunctional model with no permitted overlap.

The expression for the flux in this case is (equation 3.3)

$$J = \frac{C_x}{\frac{1}{[E_1].c_{1A} + [E_3].c_{3A}} + \frac{1}{[E_1].c_{1B} + [E_2].c_{2B} + [E_3].c_{3B}} + \frac{1}{[E_2].c_{2C}} + \frac{1}{[E_1].c_{1D}} + \frac{1}{[E_2].c_{2E} + [E_3].c_{3E}}}$$

The terms in the denominator of 3.3 are grouped according to the metabolic step they catalyse. Each term is an activity with component parts situated on different physical catalytic species specified by distinct genes. Just as in the monofunctional case, should an activity be reduced to zero, there is a block in the pathway. A complete loss of E_1's contribution to the first activity (towards reaction A) may or may not decrease flux (it depends on the effects on the contribution of E_1 to the other activities), a similar loss for reaction D reduces the flux to zero and is thus, as far as the model is concerned, lethal.

The situation is akin to a cell with a number of isozymes (or even allozymes) present. An assay may uncover an 'enzyme' with a measurable V_{max} but when a reciprocal plot to calculate the K_M is attempted the points are found not to fall on a straight line (the saturation kinetics of the different species being generally different), and extraction of the 'enzyme', or separation of the proteins of the cell by two-dimensional gel electrophoresis, instead of yielding a single catalytic species, reveals a number of independent activities. Mutants lacking one or other isozyme may, like heterozygotes carrying a null allozyme (Kacser and Burns, 1981), be relatively unaffected phenotypically (e.g. Goodman, Newton and Stuber, 1981).

It is now readily seen that equation 2.4a still represents the general case of equation 3.1, where the summation is over the n activities. As in the multifunctional case with the non-overlap assumption, the expression for the

growth rate involves replacing the 1 in the numerator of each term with the sum of the concentrations of the s proto-enzymes present in the cell. Equivalently, gene numbers may be used in the equation in the place of proto-enzyme concentrations, with c_i defined according to equation 2.12 to incorporate the relative molecular weights of the gene products. It is this formulation which shall be used from now on.

A Simple Model of Enzyme-Substrate Interaction

Having adopted the position that each proto-enzyme may catalyse several metabolic steps, and each metabolic step may be catalysed by several proto-enzymes, the necessity for some kind of representation of enzyme-substrate interaction in the model becomes clear. This requirement was also alluded to when the question of the effects of mutation on the catalytic functions of a proto-enzyme was introduced in the previous chapter. The effect was dictated to be an increase in one of the functions and an equal relative decrease in all the others. In general this is not a very satisfactory view of mutation. For example, it allows infinite improvement in any function, initially rather expensive in terms of the reduction in catalytic activity at other steps but eventually without cost (as all other functions approach zero). In addition to prohibiting mutations with differential relative affects on functions the assumption does not accomodate mutations which impair all of a proto-enzyme's activities.

The conclusion of Kacser and Beeby (1984) that monofunctional enzymes will inevitably evolve in growth-rate selected cells of the kinetic structure described rests critically on the ideas of complementarity previously discussed. Chemical species differ in their physical properties, and an enzyme active site cannot be engineered to be complementary to all of its possible substrates. High complementarity entails discrimination against many potential substrates, and is necessary for high rates of catalysis. To investigate the evolutionary behaviour of early cells, this notion must be explicitly represented.

The requirement here is to give a qualitative picture of what specificity for cells of the model consists in, and not to attempt an exhaustive chemical description of enzyme catalysis. The representation should give the flavour of the argument without demanding that extensive calculations be performed when the cells are "brought to life" *in numero*. It certainly is possible to give descriptions of the binding of substrates to catalysts enabling computer simulations of catalysis to be performed. A notable example of such studies is that by Wright *et al.* (1985) on zeolitic catalysts. The program they used, in simulating the binding of pyridine in zeolite-L active sites, calculated the miminum energy position using van der Waals and electrostatic interactions, but ignored covalent contributions (believed to be small). Their results were in very good agreement with the observed position obtained from neutron diffraction. Zeolitic catalysts, like enzymes, display complementarity to their substrates.

In the present work, it would be excessive to generate coordinates for all the atoms in each enzyme active site, and in each substrate, so that binding and catalysis can be studied. The compounds concerned have not been identified, and introducing such detail would only prejudice the abstract model. Of course, the computational costs would be prohibitively high. Instead, a simplified model, which might be described as the "box-brick" theory of catalysis shall be used.

Enzyme active sites are invariably found to be pockets or clefts in the enzyme surface into which substrates can bind. In the present work, this pocket shall be represented as a "box". The box/cleft is a hollow object with three

dimensions: a length (x), breadth (y) and depth (z). Substrates may diffuse into the active site, if they are not sterically excluded. Substrates, too, are objects in three dimensions, but they are solid ("bricks"). According to the model, any substrate which is not excluded from an active site by virtue of any of its dimensions being too large will react with the site to generate a transition state. The rate of catalysis depends on the binding energy of the transition-state with the active site.

Given the x, y, and z dimensions of active site and transition state the binding energy between them can be calculated using standard equations yielding the interaction energies between two species at a given interatomic distance. These distances are calculated by subtracting the respective dimensions of the transition state from the active site. Only van der Waals forces will be explicitly considered. The relative rates of catalysis will be directly proportional to the relative binding energies, in accordance with equation 2.21.

The particular function for the interaction energy that will be used is the 6-12 form of the Lennard-Jones formula, widely used in studies with proteins (Schulz and Schirmer, 1979). This has the form:

$$E_x = \frac{A}{R_x^{12}} - \frac{B}{R_x^{6}} \qquad (3.4)$$

where R_x is the interatomic distance in the x dimension and A and B are constants. There is a corresponding term for y, and z, and the total binding energy, E_m, is

$$E_m = 2 \left[E_x + E_y + E_z \right]$$

as there are two contacts per axis.

The Lennard-Jones function (3.4) is drawn in figure 3.1 for A=1.0 and B=10.0. Inspection of the figure shows that there is a distance ($R_m = [2A/B]^{1/6}$) giving maximum binding energy ($E_m = -B^2/4A$). Closer contact soon gives high repulsion, and longer distances involve lower interaction energies.

In the current work the interaction energy, $E_m^{\ddagger}$, for the transition state of each reaction will be treated as though it were $\Delta G_m^{\ddagger}$ (in other words, the binding energy of the undistorted substrate is always zero and the net gain in binding energy in forming the transition state from the substrate is simply the total binding energy).

Given a description of the metabolic map (a specification of its transition states as sets of $\langle x,y,z \rangle$ values), and the initial composition of the cells in the population in terms of their proto-enzymes and the numbers of genes coding for them, it is possible to give an account of mutation which avoids most of the arbitrariness previously described. Mutation alters the dimensions of protein active sites, van der Waals interactions then settle what happens to the metabolism of the cell. The model now expresses the physical constraints on the system. Which proto-enzymes contribute to which activities, and what the effect of alteration to a given active site has on these relative contributions, are determined by a simple physical view of what enzymes are and how they act.

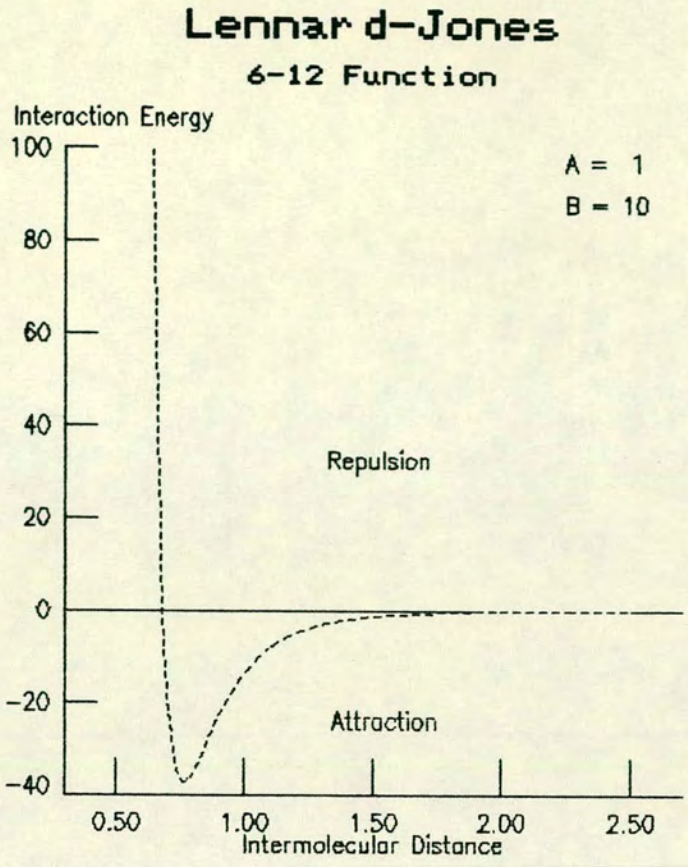


Figure 3.1

A plot of the Lennard-Jones equation:

$$E = A/R^{12} - B/R^{6}$$

used to describe the interaction energy between two uncharged molecules as a function of the separation between them.

Efficiency

It has been argued that the assumption that early enzymes displayed broad specificity made overlap between their activities inevitable. One pleasant consequence of adopting the "box-brick" theory of catalysis is that the consequential nature of overlap falls out naturally: given a set of transition states and a set of enzymes, it is possible to ask of the system what the specificities and overlaps in enzyme activities are, rather than dictate them. Nevertheless, the set of metabolites do have to be given, and the possible reactions linking them are imposed on the system by the single pathway of unimolecular transformations that has been chosen in the discussion thus far.

The arguments which lead to the conclusion that overlap in enzyme activities in the early cell would have been inevitable support equally a second conclusion: there would have been reactions involving substrate-product pairs which would not have been linked to the biosynthetic activity of the cell (i.e. which were purely wasteful). There would thus have been direct pressure on the early cells to evolve discriminatory enzymes which would avoid wasteful diversion of raw materials into unusable end products.

In order to examine the effect of such selection on the rate and direction of evolution of catalysts in systems with and without such waste, a variation on the initial model incorporating a wasteful pathway will be examined.

The simplest metabolic map which contains a flux to growth and a wasteful flux is a branched pathway of unimolecular transformations as shown in

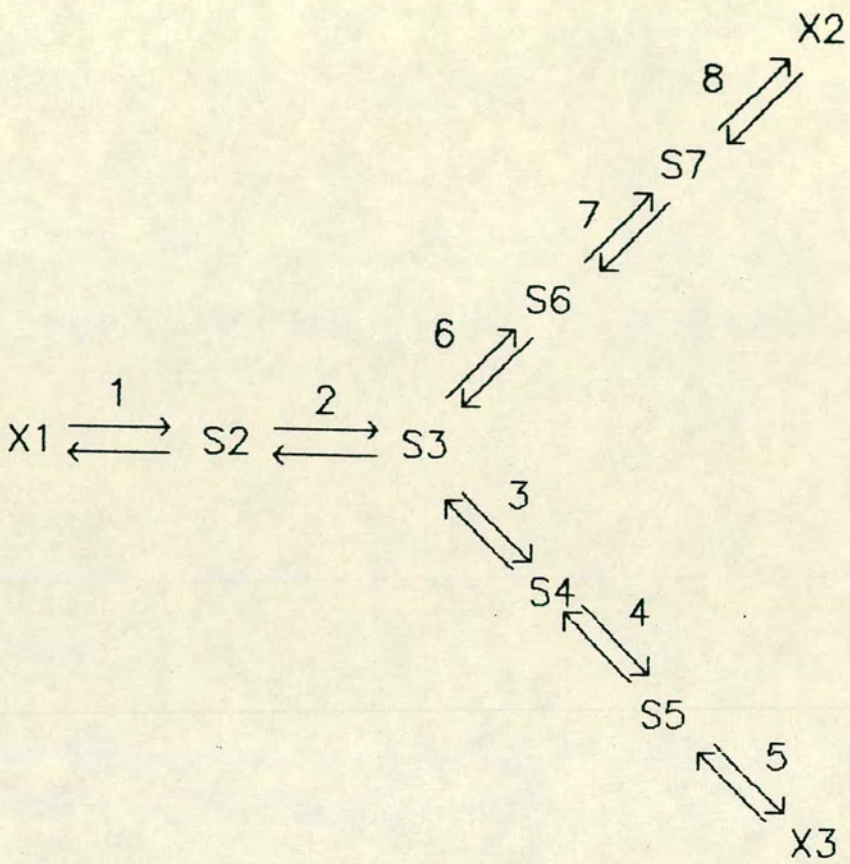


Figure 3.2

A branched pathway of reversible monomolecular reactions obeying first-order kinetics. Eight catalytic activities (numbered 1 to 8) support the fluxes from X_1 to X_2 and X_3. The flux from X_1 to S_3 is equal to the sum of the fluxes from S_3 to X_2 and X_3.

figure 3.2. Some of the characteristics of such a pathway have been examined by Keightley (1982) and Kacser (1983). The control characteristics of such a system illustrate nicely some of the features which begin to emerge as one begins to consider metabolic pathways of increasing complexity.

Like the unbranched chain of linear reactions, the system shown in figure 3.2 will, for any given set of (fixed) external metabolite concentrations, reach a steady state with time-invariant flux and intermediate substrate pools. For the linear case, one can write down an explicit formula for the flux in each arm of the system in terms of the enzyme concentrations and kinetic parameters. In the current context the fluxes to X_m and X_n are of interest. There are n activities catalysing the n metabolic steps of the system. The first 1 activities are in the common branch of the system, activities $1+1$ to m are in the branch to X_m, and the remaining ($m+1$ to n) are in the branch to X_n. The common substrate for the two competing branches is S_1 and the equilibrium constants between this metabolite and the three external species will be denoted by K_{11}, K_{1m} and K_{1n} respectively. These, and the equilibrium constants and concentrations of the external metabolites will be incorporated into environmental constants thus

$$
\begin{aligned}
C_{m1} &= ([X_1] - [X_m]/K_{1m}) \\
C_{nm} &= ([X_m]/K_{1n} - [X_m]/K_{1m}) \\
C_{mn} &= ([X_n]/K_{1m} - [X_n]/K_{1n})
\end{aligned}
$$

The flux to X_m is then given by

$$
J_m = \frac{\dfrac{C_{m1}}{(\sum_{i=1}^{1} 1/n_i c_i)(\sum_{j=1+1}^{m} 1/n_j c_j)} + \dfrac{C_{nm}}{(\sum_{j=1+1}^{m} 1/n_j c_j)(\sum_{k=m+1}^{n} 1/n_k c_k)}}{1/(\sum_{j=1+1}^{m} 1/n_j c_j) + 1/((\sum_{i=1}^{1} 1/n_i c_i)K_{11}) + 1/(\sum_{k=m+1}^{n} 1/n_k c_k)} \qquad (3.5)
$$

The alternate branch flux to X_n has a symmetrically corresponding expression involving $C_{m,}$ and C_{mn}. It can be seen that equation 3.5 is somewhat lengthier than equation 2.4. The growth rate can be calculated using equation 2.7 where we define J_m to be the flux to growth (and J_n to be the wasteful flux):

$$G = J_m / \sum_{i=1}^{n} [E_i]$$

In the simple unbranched chain so far considered the sum of all flux control coefficients must be equal to one, and as there cannot be a negative coefficient in such a pathway, the range of possible values for each coefficient falls in the range zero to one. In the branched chain the summation theorem still applies (the sum of all control coefficients relating to a flux must still be unity), but each enzyme has control coefficients relating the effect of infinitesimal changes in its concentration *on each* of the three "subpaths" in the system. In this, and more complex cases, the individual values are not constrained to the range just given. Enzymes in each of the output branches, for example, have negative coefficients with respect to the flux in the other output branch: that is to say, increasing their concentration decreases the flux in the alternate path. Keightley (1982) has shown that such coefficients may approach $-\infty$. Less obviously, coefficients of enzymes in the common branch with respect to the outputs may adopt any non-negative value. Surprising properties of this kind which are found to apply to non-linear sytems with the same map help bolster confidence in the use of linear models to investigate some of the properties of more complex natural systems.

Proto-enzymes of early cells possessing such a kinetic organisation would probably contribute to activities of more than one branch, and the elimination of the wasteful branch could not be achieved by simply deleting genes coding for proto-enzymes which contribute to the wasteful branch. Such deletions would require activities contributing to growth to be lost, and may thus result in the complete loss of an essential activity, which would be lethal. Intuitively, one feels that this constraint will increase selection favouring the evolution of high activity enzymes that can discriminate against unwanted substrates.

The evolutionary behaviour of populations comprised of cells with the kinetic organisation described so far will be investigated by direct simulation. The design and implementation of a software representation of the model will now be described.

Chapter 4

Simulation and Evolution

Scientific Computation

This chapter shall explore the use of computer simulation in scientific work, and address the question of its appropriateness in the present context. The view to be presented here is that computation is fundamental to our understanding of the universe; that for many complex systems, including (indeed, especially) organisms, no analytical description will ever be adequate and simulation is unavoidable; that any physically realizable system can be simulated on a universal computing machine; and that our understanding of complex phenomena like biological and cosmological evolution, and organismic development, shall ultimately be limited by our ability to simulate them.

Church (1936) and Turing (1936) proposed that there are ineluctable limits on the kinds of problems that can be computed in any effective procedure or algorithm. Turing gave the specification of a deterministic universal computing machine that, according to what has come to be called the Church-Turing hypothesis, can compute any function "which would naturally be regarded as computable". Modern digital computers are essentially deterministic universal computing machines in this sense (although they do not have an infinite memory like the infinite tape of Turing's machine). No improvement in the construction of computing machines, if this hypothesis is true, can extend the set of computable functions beyond those (the partially recursive functions) which can be computed by Turing's simple machine.

Various interpretations of the Church-Turing hypothesis are entertainingly discussed by Hofstadter (1979). In the 1980's, however, a new interpretation

has arisen, and it is this which potentially places the hypothesis at the centre of physical science.

According to Conrad (1985), "The strongest interpretation is that *any physically realizable system or process must be effectively computable*" [italics in original].

Deutsch (1985) proposes

> *to reinterpret Turing's 'functions which would naturally be regarded as computable' as functions which may in principle be computed by a real physical system. For it would surely be hard to regard a function 'naturally' as computable if it could not be computed in Nature, and conversely. To this end I shall define the notion of* 'perfect simulation'. *A computing machine* $\mathcal{M}$ *is capable of perfectly simulating a physical system* $\mathcal{S}$ *... if there exists a program* $\pi(\mathcal{S})$ *for* $\mathcal{M}$ *that renders* $\mathcal{M}$ *computationally equivalent to* $\mathcal{S}$ *... In other words,* $\pi(\mathcal{S})$ *converts* $\mathcal{M}$ *into a 'black box' functionally indistinguishable from* $\mathcal{S}$.

With this background Deutsch goes on to state what he calls the Church-Turing *principle* (as opposed to hypothesis) in similar terms to Conrad:

> *Every finitely realizable physical system can be perfectly simulated by a universal model computing machine operating by finite means.*

Both Conrad and Deutsch give the same argument in support of this principle. The argument rests on the assumption that any physically realizable process can be admitted as a computational primitive. So if there were a physically realizable system the dynamics of which could not be simulated by a universal computing machine, then that system could be used to admit a new primitive which would allow the set of computable functions to be extended. As, by hypothesis, no such extension is possible the Church-Turing thesis implies

that all physically realizable systems can be simulated by a Turing machine. The premise that the Church-Turing thesis is true can therefore be used to deduce that all physical processes are governed by algorithmic laws, and these algorithms can be communicated (if only we are clever enough to discover them) to a universal computing machine and used to simulate any finite physical system.

In fact, the Turing machine is a classical deterministic machine with a finite input and a set of components each of which can be in only one of a finite number of discrete states (usually 0 or 1). Dynamical systems in classical physics and in biology are continuous, and so the Turing machine can only approximate their behaviour (in principle to arbitrary precision but see the section on limits of biological simulation). Being a deterministic system, it cannot generate truly stochastic events (either in classical or quantum physical systems). Deutsch (1985) overcomes these difficulties by giving the specification of a universal quantum computer which meets the full requirements of the Church-Turing principle as he states it.

The conclusion, then, is that Nature is algorithmic and natural systems can be simulated by universal computing machines executing appropriate programs. Existing computers will be regarded throughout this discussion as adequate approximations to such universal machines for the purpose of simulating the biological systems with which we are concerned.

The Church-Turing principle, if correct, surely represents one of the most fundamental insights we have into the way the world works. For example, the

discovery of computable ordinary differential equations with no computable solutions (Pour-El & Richards, 1979, 1981) leads to an immediate prediction: no physical process will be governed by laws described by such an equation for such a process could not be simulated. I confess to being unsure of how to test this prediction!

None of the foregoing establishes that computer simulation is an effective tool in studying nature. It is not difficult to argue that the hard part of scientific endeavour is finding the 'programs'. Before one can simulate a system one needs a model of it, and once the model is forthcoming it may be that the simulation is no longer necessary. Why simulate?

Nonlinear Systems, Irreducible Systems

> *Both mathematically and physically, linear equations are the* exception *rather than the rule. Indeed, using a term like nonlinear science is, as the noted pioneer in experimental mathematics Stanislaw M. Ulam has observed, like referring to the bulk of zoology as the study of non-elephant animals.* (Campbell *et al.*, 1985).

Traditionally, much scientific computation is simply number crunching. Models in science tend to be mathematical descriptions of the relationships of measurable quantities (numbers) such as angular momentum and mass or concentration and flux. The development, testing and acceptance of such models has often involved relatively little computation and no simulation: analytical techniques have sufficed to deduce consequences which could be tested empirically in the laboratory against simple calculations. Much of the early progress in the study of multi-enzyme systems was achieved in this way.

Analytical techniques by and large depend on the model under study being *linear*. As Campbell *et al.* (1985) express it in their review of experimental mathematics

> *Linear equations are special in that any two solutions can be added together to form a new solution. As a consequence, there exist established analytical methods for solving any linear system, regardless of its complexity. ... In contrast, two solutions to a nonlinear system cannot be added together to form a new solution. Nonlinear systems must be treated in their full complexity. It is therefore not surprising that no general analytic approach exists for solving them.*

Without analytical methods, the consequences of nonlinear models are investigated by numerical solution of particular cases. Although such

solutions could be produced by hand (as, indeed, can any computation), the use of computers is generally the only practical approach.

The execution on a computer of a program embodying the dynamics of a real system is entirely analogous to performing an experiment. Henrik Kacser calls such executions "experiments *in numero*": in a general sense they are simulations. Their purpose is to tell us something about the consequences of models that can be used in testing them, and also (Kacser & Burns, 1968) to allow us to instruct ourselves on what our models mean.

So part of the case for using computer software in experimental science is to allow treatment of systems for which general analytic treatment is not possible. In this way, it is possible to investigate the behaviour of the set of equations which constitute the model. This by no means exhausts the case, however. A general presentation of the contribution of computer studies to experimental science and mathematics is presented in reviews by Wolfram (1984a) and Campbell *et al.* (1985). Wolfram points out that entities in the computer are not restricted to those observed in nature, or bound by physical law. Using a computer, one can investigate the hypothetical behaviour of magnetic monopoles (so far undetected in nature), or define and use new operations for which there is no notation in conventional mathematics (such as reversing the sequence of digits in a number). The computer thus becomes a tool allowing hypothetical universes to be explored by simulation. For the present purpose, the most important assertion is this: there exist simple rule-governed systems for which, even in principle, the only investigative procedure possible is direct simulation.

It has been stated that any physical process can be represented as a computation, and so can be simulated. One could simulate the fall of an object under gravity by computing its position at successive time intervals, for example. Such a simulation is not, for the simple case where air resistance is negligible or constant, the most efficient method of answering the question, "where will the object be at time *t?*". *One can answer this* without calculating its position at times *t-3*, *t-2*, *t-1*, ... There is an equation yielding the position of the object at any time given its initial position and the acceleration due to gravity. The application of physical laws, usually in the form of differential equations, might be thought to generally allow questions of this kind to be answered without simulating the entire evolution of the system.

Wolfram (1984b, 1984a) argues that this is not so. His work with simple mathematical entities (cellular automata) which evolve according to simple transition rules (the game called 'Life' is an example of such an entity) leads to the surprising conclusion that the evolution of such systems is *computationally irreducible* and there exists no set of equations that succinctly describe it. (The argument depends on the possibility of constructing such systems to be universal computers). For such systems, to answer the question "What is the system state at time *t*?" one has to directly simulate the evolution of the system through all preceding time points.

The possibility is now raised that there are processes in nature which cannot be described as a set of (nonlinear) equations that can be solved by successive numerical approximation for particular cases. Algorithmic

descriptions of the behaviour of such systems will, however, allow experimental simulation to be used to study them. It is possible that many outstanding problems in developmental and evolutionary biology may be of this kind.

Evolutionary Laws and Particular Cases

In the Introduction, I expressed the opinion that the history of life could easily have been different than it is: that evolution is contingent. The systems concerned are extremely complex, and their evolution is affected by many factors. Much of the difficulty in providing satisfactory general theories of evolutionary change stem from our inability to apply such theories in a testable manner to particular evolutionary cases. More than anything, it is this which has created so much controversy over the modern synthetic theory of evolution. In constructing models of particular cases to which general evolutionary theories can be applied, the amount of information to be incorporated must be very large, and the use of computers is increasingly necessary.

There is a considerable degree of disagreement about what is, and what is not, encompassed by the synthetic theory (see Gould & Eldredge, 1986; and the Prelude to Brooks & Wiley, 1986 for two recent discourses on this). Concern over its adequacy does not simply result from such disagreements. Dissatisfaction with the synthesis is becoming more and more widespread, but agreement about what is to replace or supplement it is lacking. There are those who believe the modern synthesis, however characterised, to be broadly correct but incomplete (e.g. Riedl, 1978; Reid, 1984; Wuketits, 1986); others regard it (usually because of its emphasis on natural selection) as without content (e.g. Rosen, 1982; Maze & Bradfield, 1982) or simply false (e.g. Spilsbury, 1974; Ho & Saunders, 1979, 1984). Brady (1979), in a penetrating discussion, argues that the theory of natural selection makes empirical

assertions but that, in any particular case (for the forseeable future) such assertions are untestable. He suggests that perhaps much work in biology can be done without it (the pattern cladists are the paradigm case of workers who have done this: Nelson & Platnick, 1981, 1984; Patterson, 1981, 1982). The untestability of selective arguments continues to generate an extensive literature (e.g. Mills & Beatty, 1979; Gould & Lewontin, 1979; Dunbar, 1982; Campbell, 1983; Sober, 1984; Jamieson, 1986; Waters, 1986).

To a large degree, the current dissatisfaction with the synthetic theory has been fuelled by the work of Eldredge & Gould (1972; Gould & Eldredge, 1977) and Stanley (1975, 1979) on punctuated equilibria. They saw the modern synthesis as founded on two main assumptions: (i) what they called *gradualism* (evolution occurs gradually, and most organismic change occurs phyletically as a result of natural selection) and (ii) what Bock (1970) calls the synthetic assumption (within population processes are the only evolutionary processes); and claimed it to be false on both counts (see Gould, 1980). Gould (1982) calls for a pluralistic, hierarchical approach to evolution emphasising selection at many levels.

Recent years have seen increasing attempts to break out of selectionist accounts of evolution altogether. Alternative theories generally emphasise that selection is a local agent and that the global changes in evolution (of which the increase in complexity is the most easily identified) require global explanations. Two main areas have been turned to to provide such global explanations: epigenetics and thermodynamics. Accounts of evolution emphasising the growth of internal constraints include Lovtrup (1974, 1984),

Riedl (1978), Reid (1984), Ho & Saunders (1979), and Webster & Goodwin (1982). Thermodynamic approaches include Saunders & Ho (1981), Wiley & Brooks (1982; Brooks & Wiley, 1986), Wicken (1980, 1984, 1986), Matsuno (1984), and Barbieri (1985). Such accounts do not seem, as the occurrence of the same names in both lists testifies, to be mutually exclusive, or even necessarily exclusive of the content (if not the philosophy) of the synthetic theory (which both Riedl and Brooks & Wiley claim to subsume, for example).

The model presented here, I believe, serves as a useful test case for what can and cannot be stated about the role of natural selection in evolution. Here we have a system that is subject to natural selection. The composition of the population, the model predicts, changes because of the variation in growth rates of the individual cells. Clearly, however, the model is predicting that the structure of the system allows it only one general direction to go in: towards higher complexity (that is, higher numbers of genes producing more specialised products). This, in a very simple way, illustrates the kind of constraints that have attracted so much attention. There is also a prediction that the system will become more canalised as it evolves. This is because the kinetic structure of the system inevitably gives rise to some buffering of the effects of enzyme activity changes. Initially, however, such changes simultaneously affect a number of activities and buffering is less effective as a result. As activities become independent of each other because of dissemination onto different gene products, the kinetic robustness of the system becomes more fully realised.

All these predictions are testable by running simulations of the system, and we shall examine the results later to see how they fare. Their failure, however, can hardly affect the conclusion that the changing composition of the population is largely due to natural selection, or that the direction of change is constrained by the history and kinetic structure of the system.

There are a very large number of natural systems with different structures and histories. Each of these systems have a unique history. It is possible to give general laws which apply to all hydrogen atoms, for such atoms form a class of which each particular atom is an instance. The history of each individual atom seems not to be important in explaining its properties (though the history of the universe probably is). This, it has been suggested (Ghiselin, 1969, 1974, 1981, 1985; Hull, 1976, 1978) is not so for biological entities. The species, *Homo sapiens*, is not a class of which this writer is a instance but an individual of which this writer is a component part. Adopting this view helps explain why it is that evolutionary theory has so often been charged with inadequacy. It is simply not going to be possible to produce a theory of evolutionary mechanism that can be directly applied to any particular taxon for each one is a unique individual. A great deal of additional information and theory will have to be introduced for each case. Biological science in general is science which must proceed by the pursuit of case studies (Rosenberg, 1985). The theory of evolution by natural selection has thus been referred to as a *hypertheory* (Wassermann, 1981) or *metatheory* (Tuomi, 1981) or a core theory which has no empirical content but that can be instantiated in particular cases to produce empirical statements (Brandon, 1980). Because of its subject matter it cannot be anything else.

Rosenberg (1985) argues that there are only two bodies of theory in biology. One of them is the theory of evolution by natural selection. The other is "such general principles of molecular biology as are free from any implicit or explicit limitation to any particular species or indeed any higher taxon of organisms restricted to this planet". The model system defined here, I would suggest, is a combination of these two bodies of theory. It represents a truly universal model of catalytic evolution that will apply to the early evolution of catalytic systems wherever they first occur.

The argument in this section is that biological science, unlike most of the physical sciences, progresses substantially by case studies. This is because its subject matter is a set of unique individuals connected, but also differentiated, by their evolutionary history. Universal features of biological entities, which Rosenberg assigns to areas of molecular biology but which obviously includes many of the kinetic features we have discussed, and the theory of evolution by natural selection, represent the universal laws of biological science in his view.

If such a view of biological science has merit then the promise of simulation must be large indeed. Simulation in biology becomes an important tool in the construction and investigation of particular cases or instances to which high-level biological theory can be applied in the same kind of way that obtaining particular solutions to analytically insoluble nonlinear equations can extend and test our understanding of those equations.

Limits of Biological Simulation and the Current Model

Computations are limited in two distinct ways. The first of these was identified by Turing in his original 1936 paper. It concerns the notion of computability. We have already encountered the idea that there exist functions that cannot be computed by a Turing machine or indeed (if the Church-Turing thesis is true) any other computing device. This notion is concerned with the possibility of writing an *effective procedure* to compute the function. Noncomputable functions are those for which no effective procedure can be given. In the previous section it was argued that not only can such functions not be computed, but no physically realizable system can have dynamics governed by such functions although certain ordinary differential equations (Pour-El & Richards, 1981) are numbered among them. It was asserted that any finite physical system can be simulated on a universal computing machine (Deutsch, 1985; Conrad, 1985).

The second kind of limitation has to do with physics (see Landauer, 1967). No physical device can be completely reliable, nor can it have an infinite number of components. Communication between components cannot occur faster than the speed of light and some modern supercomputers are already becoming limited in computational speed by this constraint (Conrad, 1985). The Church-Turing hypothesis gives no regard to such limitations, so it must be remembered that there is no guarantee that a particular computable function can, in fact, be computed. It is one thing to say that it is possible to calculate π to arbitrary precision, it is quite another to calculate it to, for example, 10^{50} places. No physical device that could ever be constructed would be able to

perform such a calculation. Such functions for which effective procedures can be defined but not physically executed (to completion) have been called *transcomputable* (Bremmermann, 1974). Mathematics, as Landauer (1967) has put it, is limited by physics.

More generally, algorithms that require prohibitive resources for their execution (usually because of running time rather than the memory requirements of the previous example) are said to be *intractable* (Garey & Johnson, 1979). It may be that the running time of an intractable problem is long because the effective procedure defined to solve it is inefficient. Much effort is expended in searching for more efficient algorithms to move problems out of the intractable domain and the possibility that nondeterministic (guessing) algorithms may be instrumental in achieving this has attracted considerable attention (see Lewis & Papadimitriou, 1978; and Garey & Johnson, 1979 for reviews).

Are there special problems with biological computation? Conrad (1974, 1983) has argued that adaptability depends on there being a dynamic process linking coding to function. This is clearly illustrated by biological systems. The genetic material of a cell is, in a sense, the blueprint of the cell but the products of that blueprint achieve their functional forms by a process of folding which depends on physical laws. The effect of this is that changes in the coding sequence, generating changes in the primary sequences of a protein, may have little effect on the tertiary structure and function of the protein. The result of a mutation, in other words, is a new gene product which will usually be very similar to the original. Where the product has a catalytic or

quasi-catalytic role the inherent buffering capacity of multi-enzyme systems reduces phenotypic effects of mutation still further.

None of this is true for a structurally programmable construct like a digital computer. Such a device can be programmed by reference to a finite users manual which does not require the user to consult extraneous material like the laws of physics. To see the consequence of this, consider the virtual machine, the Pascal programming language. Any algorithm can be expressed in Pascal. If one takes a Pascal program that correctly expresses some algorithm, and alters it at a single place so as to produce a second legal program, what is the effect? It is easy to show that one cannot define an effective procedure to answer that question. However, any programmer who has altered any program will know that the effects of changes will almost invariably lead to the program crashing when execution is attempted. In general, successful alterations to a program require changes at many places. There is no corresponding continuity of form and function (Cairns-Smith, 1971) in Pascal or any structurally programmable machine.

Computing machines, therefore, are not adaptable systems. By hypothesis, however, they can simulate any physically realizable system, and biological systems are adaptable and are physically realized. Therefore, computing machines can simulate adaptable entities. The question is, can they do so efficiently? Conrad (1985) proposes a **Trade-Off Principle**:

> *A system cannot at the same time be effectively programmable, amenable to evolution by variation and selection, and computationally efficient.*

and argues (1974, 1983, 1985) that the answer is no.

In the current work, the evolution of a population of cells is to be simulated. Most models and computer simulations of population evolution involve the use of selection coefficients and gene frequencies. The individual organisms do not require to be represented. The frequencies of genotypes may be calculated from the gene frequencies using assumptions about linkage between the genes, what the breeding structure of the population is, whether individuals are haploid or diploid and so on. The problems that Conrad extensively discusses do not arise in this approach.

For the model of catalytic evolution presented in the previous chapters this cannot be the approach. The genes are not merely symbols to which we ascribe selection coefficients. They specificy the structure of enzymes which have to be embedded in a multi-enzyme system so that the relative fitness of that system (organism) can be calculated. This, in itself, does not prohibit, for example, a simple infinite population model working with gene frequencies. This approach has been used in the study of multi-enzyme systems by Keightley (1982). The present model, however, also requires to represent mutation, duplication and deletion events. Mutational events will continually generate new genes, and these new genes will themselves be subject to mutation, duplication and deletion.

An infinite population model does not seem appropriate when the model has an explicit representation of gene structure, and is addressing the question of what the change in structure is going to be when selection for growth rate is applied. After all, there would seem to be an infinite number of genes (constraints could be imposed, but how to choose them?), an infinite range of

dosages for each gene per organism (within the limits set by arbitrary maximum genome sizes and complexities) and mutational frequencies from each gene to each other gene (many being close to zero, of course) and an infinite number of fitnesses to calculate.

The approach taken is much simpler. Construct a finite population of cells with a given genotype and directly simulate their evolution by explicitly storing each cell in the population with its genotype and enzymes. As cells grow and divide, and genes are mutated, duplicated or deleted, the composition of the population will change. They will adapt so as to maximise their growth rates given the range of internal and external metabolites present. This direct representation of the adaptation of the population will require considerable machine resources to implement.

Chapter 5

Implementing the Model

Programming Languages and Scientific Computation

Nature is governed by algorithmic laws. In the previous chapter, it was argued that the evolution of any system represents a computation. Scientific models manipulate numbers and symbols as abstractions of physical systems according to rules which are abstractions of physical law. Scientific computation involves expressing these models as programs for execution on a digital computer.

To execute a program based on a model of some physical process is to perform an experiment on that model system. Often such experiments amount to a single 'observation' corresponding to a single solution of a particular (set of) equation(s). This is not normally regarded as a simulation, for successive states of the system are not represented.

It is fair to say that much scientific computation is 'merely' number-crunching. Often very large amounts of processor time are required. It may be that the text of the program written by the researcher is relatively small, with most of the work being done by precompiled mathematical routines from a mathematical library (such as that by the National Algorithm Group - NAG). This is possible because most scientific models call upon frequently used mathematical operations like numerical integration or solving a system of linear equations. Often most of the user-written code is concerned with the input and output of data.

Number crunching has traditionally been done in FORTRAN. The language gives extensive support to numerical calculation, and allows the programmer to work with 'double precision' floating point reals (which give, if the IEEE standard format is used by the hardware, slightly more than 15 significant decimal digits). This degree of precision is essential over a long calculation if the number of significant figures is not to be completely eroded by cumulative rounding errors.

Simulation has additional requirements. An essential one is access to good random number generators, for simulations will often require to model stochastic events. Such generators are indeed available in FORTRAN. In general, however, the language is not an appropriate choice for simulating the evolution of a physical system. The entities being modelled in a simulation must be somehow represented in the software, and their abstract properties defined. This is difficult in FORTRAN which has only a very weak concept of *type*.

All high-level programming languages have a typing system (for a review, see Tennent, 1981; Ghezzi and Jazayeri, 1982; or Horowitz, 1983). In imperative programming languages, in which virtually all scientific programming is done, programs manipulate *variables*. A variable is an abstraction of a memory cell: it is a named location in memory. Variables have a type, and the type of a variable is the set of values it may have and the operations that may be applied to those values. Each programming language provides a set of primitive types and the user may create and modify variables of these types according to the predefined operations supplied. FORTRAN 77 provides the

predefined types: INTEGER, REAL, LOGICAL, CHARACTER, DOUBLE PRECISION, and COMPLEX. For simulation, these predefined types and operations are not sufficient. Modern programming languages provide facilities for users to specify their own types. When such a language is strongly typed (that is, does not implicitly convert values of one type into values of another) this facilitates the mapping of abstractions onto the language and promotes reliability.

Consider the current model. A flux can be represented by a real number, so can an enzyme concentration. FORTRAN has no rules to prevent the researcher from adding a flux to an enzyme concentration, or from assigning the result of dividing a flux by the sum of all the enzyme concentrations to a K_M rather than a growth rate variable. As programs become increasingly large such errors become more likely and extensive testing becomes required before one can display any confidence (and never certainty) that mistakes of this kind and other (for example, typographical) errors leading to nonsensical outputs are not present in the program text. A strongly typed language, on the other hand, can catch most such errors without ever executing the program when (at compile-time as the jargon goes) the program text is translated into the machine language of the computer on which it will run.

The independent compilation facilities of FORTRAN, allowing the provision of libraries of subroutines which the user can call upon, contributes to the possibility of runtime errors, for there are no checks that a call of a library subroutine expecting a REAL parameter, say, is not passed a LOGICAL or CHARACTER value instead, so that garbage values are used. In scientific

computation, where so much use is made of mathematical and statistical libraries, compile-time checks on the use of library units are clearly desirable.

For simulation purposes, FORTRAN is less widely used. Generally the number of entities to be represented in a simulation depends on events which occur during the simulation. In discrete event simulation a list of events, ordered according to simulation time, is kept. Events are generated, which may add or remove elements from the events-list. Because the list may be of variable length the memory requirements of the program may vary during execution. To cater for such variation, many programming languages provide *pointers* which can be used to dynamically (that is, at run-time) create objects on a so-called *heap* (an area of memory which can contract or increase in size as the program requires). FORTRAN does not provide such a facility, however, and the total space a program requires must be reserved before it is executed. As there is no way to dynamically appropriate more space as the program executes one must reserve sufficient memory for the largest possible run; a wasteful solution in general.

The particular requirements for simulation were recognised early in the evolution of programming languages and resulted in the development of Simula 67 in the late 1960's to meet the requirements for discrete event simulation work. It provided the ability to define *classes* of objects with associated operations, and to create coroutines which could be used to express parallelism. Simula is now, as Pooley (1985), in an examination of the requirements of a modern simulation programming language states, "out of date".

It carries many unwieldy legacies from ALGOL 60 (on which it was based) and lacks the algorithmic richness and facilities for creating data types possessed by more recent languages like Pascal and its descendants, Modula-2 and Ada™. It seems unlikely, now that most of Simula's better features are represented in modern general purpose languages, that there is a real requirement for a replacement language for Simula that is specifically simulation-oriented though there is still considerable room for expansion in terms of these facilities, as Pooley (1985) describes.

When the implementation of the simulation is discussed later in this chapter, the importance of information hiding will be stressed. At this point it is worth emphasising that the entities being modelled in a simulation represent objects with certain properties. For reasons of security and ease of understanding, the specification of these properties should be kept distinct from how those properties are achieved or represented in the program. Implementation details for a given entity should be hidden, for in the world view of the model, the properties of entities do not depend on them and neither should any user program operating on that entity. The particular problems associated with this model and this simulation will be described as they are encountered.

The general requirements for the current work, then, with respect to the programming language in which the simulation is to be implemented, is for a strongly typed language with good facilities for numerical work, with separate (not merely independent) compilation facilities, support for information

hiding, a rich set of algorithmic primitives, facilities for defining records and complex data structures, pointers, and possibly a model of concurrency.

My first intention, when embarking on the simulation, was to attempt an implementation in Pascal despite its deficiencies with respect to some of these requirements. Many Pascal systems provide double precision reals by default (though the standard language does not allow precision to be specified), and nonstandard facilities for separate compilation are also common. Pascal supports user-defined types and provides records and pointers. Good Pascal compilers, generating code as efficient as most FORTRAN systems, are available on various service machines at Edinburgh university.

Pascal was designed as a small teaching language by Niklaus Wirth in 1971. It is in many ways an elegant language, and has, somewhat surprisingly, become very popular as a vehicle for applications programming. An ISO standard definition of the language now exists (Jensen and Wirth, 1985). Unfortunately, its suitability for serious applications depends in large part on nonstandard features as the previous paragraph illustrates. Furthermore, for numeric values, it is not really any more strongly typed than FORTRAN: all real numbers are treated as the same type and there is no way to express the distinctness of fluxes and catalytic constants, for example. Similar problems occur with integers though Pascal will allow the user to specify subranges of integer and raise a runtime error if this range is violated. Programmers often use compiler directives to suppress runtime checks of this kind as programs run more slowly because of them. Pascal has no support for

separating specification from implementation, so its information hiding facilities are poor. Finally, it has no model of concurrency.

Modula (Wirth, 1977), and its later incarnation as Modula-2 (Wirth, 1982; 1983; 1985) was based on Pascal, and incorporated many ideas from the development of programming language theory in the 1970's. In particular, separate compilation facilities, separation of specification from implementation, and fairly simple coroutines to represent concurrent processes were built into the language which was intended for serious systems programming. At around the same time that Modula was being developed, the US Department of Defense undertook an extensive analysis of its software needs that led to the commissioning of a new high level language for real-time programming. This language (Ichbiah, *et al.*, 1979), too, was based on Pascal, and on acceptance of the final language definition by the DoD in 1982 was called Ada. The language became an ANSI standard in 1983 (ARM, 1983). These very recent languages were not available within the university, when, in 1982, I began seriously considering computer simulation of the model.

This changed when in 1984 I was fortunate enough to be appointed to give Ada support to the UK academic community on a Computer Board funded post based in the Edinburgh Regional Computing Centre. With access to the validated Rolm/Data General Ada compiler, the possibility of an implementation of the simulation in a powerful new language became possible and this chapter describes this implementation.

Object Oriented Programming and Abstract Data Types

The entities to be represented in the simulation comprise a population of cells. Each cell has a genotype composed of a number of genes with particular dosages. The genes specify protoenzymes which have concentrations that depend on the gene dosage and catalytic activities (towards particular substrates) that depend on the gene sequence. The cells increase in mass at a certain rate due to the flux to growth (determined by the concentrations and activities of the protoenzymes) and this determines how frequently they divide. Gene dosages alter by duplication or deletion, and gene sequences alter by mutation, with a given frequency.

The present chapter is concerned with how these properties were specified in the software, how they were implemented, why certain decisions were taken and reversed, and what problems were encountered along the way. This discussion will by no means be exhaustive, or even cover every library unit in the program. I firmly believe, however, that the time has come for the principles of sound software engineering to be applied to scientific programming because of the increasing size and complexity of the programs required, and that this, for the next decade at least, means implementing scientific work in an imperative language supporting those principles and providing good facilities for numerical work. I shall not discuss here the relative merits of different languages beyond the few comments made in the previous section, but would urge that Ada should always be considered as the language of first choice. Some of the reasons for this advocacy will become clear as the chapter develops.

The approach I shall take to the specification of the software system will be object-oriented. Object oriented programming has its beginnings with Simula and with the Smalltalk environment, and has been extensively developed for use with Ada (Booch, 1983; Buhr, 1984; Wiener & Sincovek, 1984). Ada does not allow the representation and properties of objects to be defined as a program is actually executing (dynamic binding), but demands that all such decisions are known when the program is translated (static binding). *Sensu strictu*, this makes it difficult to justify describing Ada as supporting object oriented programming without building language extensions (see, for example, Cox, 1986; Cardelli & Wegner, 1985). The data abstraction approach to program design (see below, and Liskov & Guttag, 1986) is very well supported in Ada, and is what is meant by "object oriented programming" in the Ada literature.

The process view of simulation (Franta, 1977), in which the system being modelled is decomposed into processes (active objects) that interact with each other, results in object-driven simulations for which an object oriented approach is well suited. This will be the approach taken for the present work. Interaction between cells, admittedly, is likely to be limited.

In the current model, the cells, genomes, genes, protoenzymes and substrates are obviously objects. These objects can be defined operationally. In Ada, it is possible to express *abstract data types* using packages (modules) exporting private types. Abstract data types are described below. A private type is a programmer defined type with values that are hidden from the user of the package (in terms of how the package writer has chosen to represent the type on the machine) but instances of which the user can manipulate using the

operations that the package *exports*. Languages like FORTRAN and Pascal do not support such a facility or lend themselves to object oriented programming.

The provision of abstract data types enables the programmer to declare types which have the same logical status as the predefined types in the language. For example, INTEGERs are whole numbers. Such numbers can be manipulated using the usual arithmetic functions. It should not be apparent to the programmer that, internally to the computer he is programming, an integer is a not a scalar but a string of zero's and one's (bits). The programmer should not be required, or indeed allowed, to directly access or change the eighth bit in a string representing an integer, for that is not an operation that is arithmetically defined for whole numbers. The internal representation of an object in a computer program should have no impact on the abstract properties of that object for any program that uses it. In most programming languages, however, if a programmer wishes to create a package for manipulating FRACTIONs, for example, it is not possible to hide from a progam using that package that a fraction is a record with a component for the numerator and a component for the denominator, and no way to stop such a user program from directly accessing and changing these components without using the arithmetic operations provided by the package.

The initial design problem, then, is to give a specification of the system in terms of the objects required and the operations upon them. This will simply be presented without any further justification. Implementation will not be discussed very extensively, except for the problem of how to represent cells.

Cells as Processes I: A Naive Approach

The intended approach to the implementation is going to be object oriented (this will be discussed more fully in later sections of this chapter), and to the simulation, process oriented. The processes in the simulation are easy to identify, they are the cells. (Evolution is not something that is going to be identified as a process for the purposes of the simulation, it is simply synonymous with the changes to the state description of the population as the simulation proceeds).

A process oriented simulation view is most easily expressed when the implementation language has some way of directly expressing processes. A process is a concurrent activity, at least notionally proceeding in parallel with other such activities.

Simula 67 and Modula-2 both provide *coroutines* to represent processes. Coroutines are modules with their own *thread of control*, and the programmer can interleave their execution. When a subroutine (or subprogram as they are now more usually called) has finished executing it returns control to the unit that called it, and any data local to that subprogram is lost. A coroutine, on the other hand, does not return but explicitly transfers control elsewhere, retaining its data and the point in its execution that it has reached. A coroutine can therefore resume when called upon at the point it left off. In both Simula and Modula it is possible to dynamically create processes (that is, the number of processes can depend on events during a simulation/program and need not be statically specified when writing the program).

The coroutine is rather a low-level construct, in that the programmer must specify when each coroutine should transfer control to another. Ada does not provide coroutines. Instead, the language represents processes by truly concurrent objects called *tasks*. On a computer with several processors, it is possible for several tasks to be physically executing in parallel. (This is not so for coroutines, the design of which assumes that there is only a single processor available). However many physical processors a machine may have, the Ada language definition requires that the effect of the execution of tasks is such that each task has its own *logical processor*, and the interleaving of task execution is performed by the runtime system automatically. The Ada model of concurrency is thus an extremely powerful one (for a review see Burns, Lister & Wellings, 1985). Ada tasks can be created dynamically using pointers.

If a process oriented view is to be adopted, the natural implementation decision is to use tasks to represent cells. Cell division will result in the generation of two new processes, and the termination of the parent process. Such an approach has its attractions and can be realised by representing a cell something like this

```
task type CELL is
   entry Genome_Is (Genetic_Endowment : in GENOME);
end CELL;
```

The entry to the cell allows information to be communicated to it, in this case, so that it may receive its genetic endowment from its parent. The body of the task defines its actions: firstly to accept its genome and then grow and divide. The frequency of cell division depends on the growth rate of the cell. Ada provides a *delay statement* that can be used to suspend a task for a

specified number of seconds. This immediately suggests the following implementation

```
task body CELL is
   My_Genes : GENOME;
begin
   accept Genome_Is (Genetic_Endowment : in GENOME) do
      My_Genes := Genetic_Endowment;
   end Genome_Is;
   delay (1.0/Growth_Rate(Catalysed_By=>My_Genes));  -- S phase
   Divide(My_Genes);                  -- creates two new cells
end CELL;
```

where the delay statement is used to delay the cell for the period required to double in mass (the reciprocal of the growth rate since the faster the growth rate the shorter the cycle length) during what corresponds to the 'synthetic' phase of the cell cycle.

The procedure, Divide, which takes the genome of the cell as a parameter, generates two new cells and passes each of them a copy of the genome. During this transmission the genome may be subject to segregational errors resulting in duplication or deletion of genes. Abstractly, any mutations that have occurred in replicating the genome will have occurred during S phase, but as this is implemented by suspending the process the Divide procedure will also introduce mutations into the transmitted genome with some probability, p. The function, Growth_Rate, takes the genome as a parameter and returns the growth rate supported by the gene products of that genome.

A simulation based on this abstraction of a cell requires only to create an initial population of one or a few cells to set evolution in motion. Adjustment of the value of the environmental constant, C_x, (see equation 2.4) will be required so that the duration of a delay is sufficiently short to run

the simulation in an acceptable time, and sufficiently long that it is significant compared with the time required to create a process, pass it its genome, and calculate its growth rate. Cumulative drift in the frequency of cell division because of the overheads of these operations can be eliminated, as long as the delay is sufficiently long, by using a clock to ensure that the actual delay compensates for the time already used in performing them. Ada provides access to such a clock in the predefined package, Calendar.

My first attempt at writing a simulation of the model in Ada was based on the ideas briefly outlined above. Such an approach takes a great deal of implementation work out of the simulators hands. The most significant deviation from a standard discrete event simulation (whether process oriented or not) is that there is no explicit events-list. The processes here, cells, generate only one event, a cell division. Cells vary in growth rate and hence in the interval from one cell division to another. In a traditional discrete event simulation each cell division would generate a record (an *event notice*) representing the time of the next division, and this notice would be inserted in the events list at the point corresponding to the time at which it is to occur. Such a simulation proceeds (once one or more initial event notices have been generated) by repeatedly removing the earliest event notice from the front of the list, updating the simulation time to that for the event, and generating two further notices (corresponding to the two daughter cells) which are inserted in the list at positions appropriate for the time that they are scheduled to divide (the two daughters may differ in growth rate because of mutation).

In the present suggested implementation, however, there is no events list required, and no event notices. The passage of time in the simulation is entirely represented by the real-time delay of the cell during its notional S phase. No list processing is required. The responsibility for the correct sequence of events is delegated to the runtime environment of the simulation program. The completion of the cell's S phase is modelled not by explicitly removing an event notice from a list, but by the automatic activation of the cell when the delay interval has elapsed. The division of the cell does not require the explicit creation and insertion of notices but is achieved by the creation of two new tasks, that are suspended for the period they require to double their mass.

Cells as Processes II: The Limits to Growth

One problem with the cells-as-tasks approach concerns the memory requirements of the simulation. There is the obvious point that as each cell division terminates one cell but creates two the memory requirements of the simulation are going to increase as the simulation progresses. This point will be returned to but is not a problem unique to the tasking approach. A traditional simulation would be faced with the same problem, simply because it really is part of the 'real world' system being modelled. Population growth will result in any population of cells running out of resources. However, in the real world, biological and chemical cycles result in the material of which cells are composed being returned to the environment once the cells die. Unfortunately, the natural return of resources to the simulation cannot be so readily assumed.

The memory area allocated to a program in Ada (and other block-structured languages) consists of a *stack* and a *heap*. The size of both of these areas varies during the execution of the program, but in different ways. The problem for the current simulation concerns the subprogram, Divide. In the outline implementation of the cell task above, the last statement is a call of this procedure with the genome of the cell passed as a parameter. Consider how this procedure might be implemented:

```
procedure Divide (Parent_Genes : in GENOME) is
   First_Daughter,
   Second_Daughter : CELL;
begin
   First_Daughter.Genome_Is( Replicate_Of(Parent_Genes) );
   Second_Daughter.Genome_Is( Replicate_Of(Parent_Genes) );
end Divide;
```

where Replicate_Of is a function that returns a copy of the genome passed to it as a parameter, but may introduce mutational or copy number changes with a certain probability.

This implementation gives rise to memory management problems. When the procedure Divide is called by the parent cell a new frame is added to its stack. The two cells declared inside the procedure themselves have stacks that will be rooted in that new frame. When each cell terminates, their stacks are removed, and when the procedure Divide terminates, it is removed from the parent stack, and the memory so released is returned for re-use. This seems to nicely achieve the recycling of memory resources intended. Unfortunately, this is not the case. The procedure cannot immediately terminate once it has communicated with the cells it has declared but must wait for both daughter cells to terminate. The parent cell (which has called the procedure) cannot, therefore, terminate until both of its daughters have divided and terminated. Worse still, the problem is recursive, applying equally to each of the daughters, which must wait for *their* daughters to divide and terminate. In other words, the first cell cannot terminate (returning the memory resources allocated for its stack) until all of its descendants have! The main program stack, bearing the stacks for each of the tasks (a so-called *cactus stack*: Burns, 1985) continues to grow and grow, with no recycling of memory at all. This bears no relationship to the biological position being simulated and seriously limits the size of any simulation.

The problem in the above solution follows from the rule in Ada that every task has a parent unit, and that such a parent unit cannot terminate until all of

its children have. Because the First_Daughter and Second_Daughter tasks were declared in the procedure Divide, that procedure was their parent and has to wait for them. This rule ensures that a stack frame cannot be popped (removed from the stack) while it still contains the roots of stacks belonging to tasks that are still executing. Clearly the code above does not meet the requirement that after a cell has divided it no longer exists.

If the parent cell, which calls the procedure Divide, is to terminate before the daughters it creates does, and these daughters are created by the execution of the procedure, then the solution must be constructed so that neither the procedure nor the parent cell are, in the Ada sense, the parent unit of the daughters. Fortunately, this is possible by creating the cells using pointers, which has the effect of rooting their stacks not in the main program stack, but on the heap. The parent unit of a task created via a pointer variable is that progam unit in which the pointer *type* (not the variable) is declared. If the pointer type to the task type, CELL, is called CELL_POINTER then the following implementation of Divide is not required to wait for the cells it creates, so that a parent cell, calling Divide, may terminate as soon as its children are born.

```
procedure Divide (Parent_Genes : in GENOME) is
   First_Daughter,
   Second_Daughter : CELL_POINTER := new CELL;
begin
   First_Daughter.Genome_Is( Replicate_Of(Parent_Genes) );
   Second_Daughter.Genome_Is( Replicate_Of(Parent_Genes) );
end Divide;
```

Although the execution of each cell now corresponds to what is desired, the memory management problem has not been solved. Memory allocated to frames placed on the stack is reclaimed automatically when the stack frame is popped.

This is why Ada requires that parent units must wait for their children, so that the working space allocated to such children is not allocated to some other process while still in use.

Objects placed on the heap, on the other hand, persist virtually independently of what happens on the stack. Automatic reclaiming of areas of the heap, when it occurs at all (many runtime environments do not have any automatic reclamation) is expensive, because it must be possible to guarantee that an object placed on the heap is not still accessible by an existing pointer variable. The only truly secure and inexpensive way of doing this is to reclaim the space only when it is impossible for such a pointer to exist: when the pointer type is no longer known (because the unit in which the type was declared has terminated).

To allow objects allocated on the heap to be returned to the environment when no longer required, languages like Pascal, Modula-2 and Ada provide a *unchecked deallocation* or *dispose* facility to enable the programmer to explicitly return them. It then becomes the programmer's responsibility to ensure that once such an object is returned there are no pointer variables that still access it (so called "dangling references"). Use of such a facility is potentially very dangerous, and Ada requires the programmer to explicitly import it to his/her program to alert the reader of its use, but is often essential if a program is not to run out of space.

The use of unchecked deallocation is, with due care, unproblematical when the object being accessed is a record or other data object. Unfortunately, it is

not at all unproblematical when the object is (or contains) a process. If a record is not accessible by (or contain) any process then there can be no conceivable damage to the program that created it if the record is destroyed and the space it filled reused. If the deallocated object is a process, however, the question of whether the process is accessible to any other process is not a criterion of safety for its storage to be reclaimed. As long as the process is executing, it clearly must have the use of its work space which must not be overwritten. For this reason, the Ada language definition requires that the programmer cannot reclaim the space allocated to any process created by a pointer. Attempts to use unchecked deallocation in this way have no effect on the task. This means that, with the task implementation given above, the growth of, and inability to reclaim space from, the program stack is simply transferred to the program heap.

To avoid this difficulty, cells must be constructed in such a way that they are reusable. The requirement for a pool of reusable tasks in systems programming when there is a requirement to dynamically create service tasks has been discussed by Burns (1985). In introducing a pool of reusable tasks, the implementation of Divide using CELL_POINTER can be retained almost unaltered, and it is simply necessary to rewrite the body of the cell task so that, once it has divided, it returns itself to a pool of cells by a call of a new procedure, Return_To_Pool. To return itself, the cell requires to know its identity. Instead of obtaining new cells by a call of the language defined allocator, **new**, a function, New_Cell, must be provided that will attempt to obtain a cell from the pool when a call to it is made, and only create a **new** cell object if the pool is empty.

```
function New_Cell return CELL;

task type CELL is
   entry Personal_Details (Identity    : in CELL_POINTER;
                           Genetic_Endowment : in GENOME);
end CELL;

procedure Divide (Parent_Genes : in GENOME) is
   First_Daughter,
   Second_Daughter : CELL_POINTER := New_Cell;
begin
   First_Daughter.Personal_Details_Are
         ( Identity=>First_Daughter,
           Genetic_Endowment=>Replicate_Of(Parent_Genes) );
   Second_Daughter.Personal_Details_Are
         ( Identity=>Second_Daughter,
           Genetic_Endowment=>Replicate_Of(Parent_Genes) );
end Divide;

task body CELL is
   My_Genes : GENOME;
   My_Name  : CELL_POINTER;
begin
   loop
      accept Personal_Details_Are
               (Identity    : in CELL_POINTER;
                Genetic_Endowment : in GENOME) do
         My_Name  := Identity;
         My_Genes := Genetic_Endowment;
      end Personal_Details_Are;
      delay 1.0/Growth_Rate(Catalysed_By=>My_Genes);
      Divide(My_Genes);
      Return_To_Pool(My_Name);
   end loop;
end CELL;
```

The notion of self reproducing Ada tasks is not new with this simulation: the idea of using them to explore a sequence of moves in a game has been examined by Lomuto (1983), though the problem of storage exhaustion was not addressed as the example was not intended as a serious piece of Ada programming. Reproducing processes are surprisingly simple to express in Ada, which is why the idea of using such an approach was almost irrestible at the initiation of the project. The naturalness of the solution, even with the irritation of

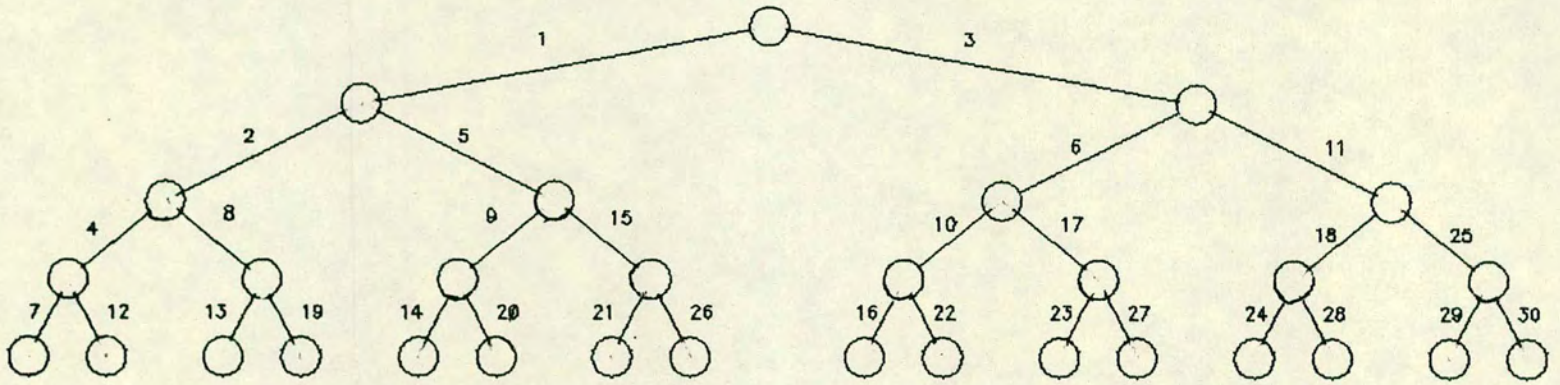
having to provide a pool of reusable cells, makes it easy to understand and implement. Issues such as monitoring the progress of the simulation, outputting the results, and terminating it, are put aside here for simplicity. They are not directly relevant to feasibility of performing the simulation at all, or to the representation of the abstract (biological) properties of the model.

Even with a reusable pool the program will of course eventually run out of space, just as any exponentially expanding population will, irrespective of how efficient in recycling it might be. The niche space is not infinite, and neither is the memory allocated to a program on a digital computer. There are two possible solutions to this problem. The first, which I chose for the first complete implementation of the model, is to prevent the population from expanding indefinitely by putting it through periodic bottlenecks. This solution was adopted partially because the runtime scheduling of tasks proved, as I should have realised, to be quite incapable of allocating tasks to the processor in a sequence consistent with the passage of time as perceived by the model. Figure 5.1 illustrates the problem. The bottlenecks, occurring every five generations, served as synchronization points for the simulation, and involved a positive selection of the four fastest cells in each fifth generation as the founders of the next.

The alternative solution is to allow the program to run out space, and then take some action. This could be made to correspond to the natural situation where, as the resources used by the population grow more scarce, members of the population fail to survive and the population size oscillates or reaches a

Figure 5.1

Scheduling of Ada Tasks: A Problem



The above diagram illustrates the order in which cells (represented by tasks) were created during a short test program. Each cell cycle was the same length and so the desired order is according to generation. In fact, some fifth generation cells were born before some third generation

steady state (see Eigen and Winkler, 1975). In Ada, an attempt to execute a statement that results in the memory allocation of the stack or heap being exceeded raises an error condition (an *exception*) known as a Storage_Error. This condition can be handled by the program (a similar error in most programming languages would result in the program crashing) by returning space to the environment before proceeding. This is doubtless the better path, but makes no allowance for the problems of figure 5.1.

Figure 5.1 illustrates that the scheduling algorithm of the runtime system cannot be expected to fairly allocate processor time to cells in accordance with the abstraction of simulation time required here. The larger the population of cells is required to be, the larger the problem. Ada has been examined seriously by a number of workers as a vehicle for expressing discrete event simulations, and various general purpose packages with quite diverse philosophical ends now exist (Lomow & Unger, 1982; Unger, Lomow & Birwistle, 1984; Inkster, Lomow & Unger, 1984; Downes & Taelleche Bosch, 1983; Bruno, 1982, 1984; Bryant, 1982; Sheppard, Friel & Reese, 1984;, Friel & Sheppard, 1985). Those which attempt to use the Ada tasking model to allow process simulation packages have all adopted a traditional events-list solution.

From the difficulties encountered in the current work, and an examination of the literature, I attempted to define a general package that would conceal the tasking element, and allow the events list to contain suspended processes of many different kinds. The result of this attempt, the implementation of which was done largely by a final year undergraduate student in the Computer Science department of Edinburgh University under my supervision, was presented to the

Ada-Europe Conference on *Ada: Managing the Transition* (Steele & Beeby, 1986). The existence of a single events list creates a bottleneck that reduces the usefulness of the tasking approach, particularly when there are a large number of processes in the simulation.

Before undertaking the attempt to produce such a general purpose process simulation package, particularly with the needs of industrial applications in mind, I had already abandoned the idea of using the tasking model to simulate metabolic evolution. Such a simulation requires a representation of a (fairly large) population of evolving cells, and the cells-as-tasks approach became untenable once I had seen the results of a simple experiment performed by Dr Alan Burns (University of Bradford) on the Data General MV4000 I was using, in preparation for his book on the Ada tasking model (Burns, 1985). I do not have his data, but have repeated his experiment, which is shown in figure 5.2. The results give a measure of the cost on the Data General/Rolm system of using the tasking model. Whenever a task is swapped onto the processor, the context of the process that has been swapped out needs to be stored, and the context of the current task mounted. This *context switching* is very expensive in time on all existing Ada implementations (though DG/Rolm do considerably worse than some others) and is intolerable for the present purposes when the costs, in view of the arguments of Conrad and others, are already likely to be very high. The use of coroutines, as provided by Simula 67 or Modula-2 would be much less prohibitive, but would not have the naturalness of the Ada solution. I decided simply to use a traditional discrete event approach, which required first a package that would implement the events list.

Figure 5.2

A Simple Program To Determine the
Cost of Task Communication

```
with Timer;
use  Timer;
procedure Task_Version is

   Size_Of_Job : constant := 2000;
   J           : INTEGER  := 0;
   task T is
      entry Add (I : in out INTEGER);
   end T;
   task body T is
   begin
      loop
         accept Add (I : in out INTEGER) do
            I := I + 1;
         end Add;
      end loop;
   end T;

begin
   Timer.Start;    -- Starts timing elapse and processor time
   loop
      T.Add(J);    -- Calls the task T to increment J
      exit when J = Size_Of_Job;
   end loop;
   Timer.Stop;
   Timer.Print_Elapsed_Time;
   Timer.Print_CPU_Time;
end Task_Version;
```

The procedure version is identical to above, but the task is replaced by a procedure, Add, with the same parameters and body as the task entry of that name. The loop has a call on the procedure rather than the task. In the actual experiment, the task body and procedure Add were compiled separately from the main program to prevent any compiler optimisation of the loop.

Timer results from the above on a Data General MV4000:

	Procedure Version	*Task Version*
Elapsed Time (seconds)	0	22
CPU Time (milliseconds)	95	17638

DG ADE 2.30 release is very inefficient for Ada tasking. Other Ada systems give factors of 20 to 50 in the time of a task entry versus procedure call.

The Package Population

What events are required to be stored in the events list of the simulation? In the procedure Divide of the cells-as-tasks implementation, replication errors occurred implicitly at cell division. The evolution of the population can be thought of a sequence of cell divisions. The division of cells are the events.

Even abandoning the tasking implementation, there is no reason to alter this view of evolution. The events list becomes a chronologically ordered list of cell divisions. Each cell division generates two new cells that will themselves divide after an interval that can be calculated from their growth rates. In effect, the events list *is* the population of cells. The first cell removed from the population at its cell division time generates two more cells which, if viable, join the population (are inserted into the list at the points corresponding to their cell division times) to await division. Then the next cell is removed from the population and the process is repeated, the population growing with each division (assuming the frequency of lethal mutations is less than 0.5 per cell division).

This behaviour can be implemented as an Ada package without worrying at all about what a cell is. Ada provides facilities to write program units that are parameterised by the type of object upon which they are to operate, and so the primitives required to support the manipulation of the events-list/population can be wrapped up into a package before making any further implementation decisions. It is possible, therefore, to avoid making any assumptions about

what a cell or genome is, and as the development of the simulation continues, it will be possible to alter decisions about how objects are to be represented without any effect on the current package whatsoever.

As this is not a tasking approach, there is no reason not to adopt the solution to the overpopulation problem discussed above, and allow the population to saturate its environment before taking any action to control it. At that point, space must be returned to the environment and the simplest way to do this is to "kill" some of the cells in the population. An operation to remove arbitrary cells from the population will therefore be included. This gives the generic package specification shown on the following page.

The package expects to be told what type is to be used for the simulation time. It will allow any type that has an ordering relation (the function "<" is explicitly asked for), though clearly a numeric type seems appropriate. An explicit Start_Time must also be given. The maximum size of the population is also a parameter, so that different instances of Population may have different maximum sizes. The member type can be any type for which assignment and equality are defined (so not a task).

Internally, the population is a doubly linked circular list of record elements, ordered by the value of SIM_TIME passed when the element is stored. A facility is provided to sample the population nondestructively by using the function, Random_Sample.

```
generic
   type SIM_TIME is private;          -- any assignable type
   Start_Time    : SIM_TIME;
   Max_Pop_Size  : POSITIVE;          -- minimum value is 1
   type MEMBER_TYPE is private;       -- any assignable type
   with function "<" (Left, Right : SIM_TIME)
      return BOOLEAN is <>; -- default: normal less-than operator

package Population is

   -- Removes that member of the population due to be restored
   -- next and returns it, advancing the clock accordingly.
   -- Raises No_Population if the population is empty.
   function Restore_Next return MEMBER_TYPE;

   -- Removes that member of the population due to be restored
   -- last and returns it.  Raises No_Population if the
   -- population is empty.
   function Restore_Last return MEMBER_TYPE;

   -- Returns the current time.
   function Sim_Clock return SIM_TIME;

   -- Returns the time that the next member of the population is
   -- due to be restored.  Raises No_Population if there is no
   -- such member.
   function Next_Time return SIM_TIME;

   -- Stores the member in the population for restoration at the
   -- SIM_TIME specified if the population size is less than
   -- Max_Pop_Size, otherwise raises Overpopulation.
   procedure Store (Member : in MEMBER_TYPE;
                    Until : in SIM_TIME);

   -- Returns the current population size.
   function Population_Size return NATURAL;

   -- Kills a random member of the population.
   -- Raises No_Population if the population is empty.
   procedure Kill_Random_Member;

   -- Samples a random member of the population nondestructively.
   -- Raises No_Population if the population is empty.
   function Random_Sample return MEMBER_TYPE;

   No_Population, Overpopulation : exception;

end Population;
```

Layers of Abstraction in the Model

Conrad (1974) argues that the evolution by natural selection of complex systems cannot be efficiently modelled on a structurally programmable computer. To saddle a simulation of such evolution, even involving fairly simple multi-enzyme systems, with the problems of context switching in the Ada tasking model is clearly perverse, however natural the solution may look.

A traditional discrete event simulation seems, therefore, the most promising approach. Simula 67 was designed to support such simulation, and gave rise also to object oriented design methods. Although it is fair to say that the present implementation is going to be object oriented, yielding a set of packages implementing abstract data types (objects with their associated operations) the particular methodology of that name developed initially by Abbott (1983) and adapted to Ada by Booch (1982) will not be used. As Rajlich (1985) has argued, this methodology (an example of what Rajlich calls the *traditional large-small paradigm*), requiring that all the package specifications be given before any implementation begins (not quite correct, but I shall let that pass), is very unforgiving of design errors. Much more suited for the present work, involving as it does a single researcher implementing an exploratory program, is an *incremental bottom-up* approach, where the lowest level package specification is written first and possibly implemented, then the next highest layer (making use of the previously written package) and so on so that the system is constructed from the bottom up. This approach is very much more forgiving of design errors. Although Rajlich's paper had not yet been published when the design effort began, this

was the path I naturally adopted. I believe the reason for this can be seen when the logical structure of the model is examined.

The model asserts that the earliest cells were selected for growth rate. A cell is a multienzyme system converting an external metabolite, X_o, into protein (algebraically treated as another external metabolite of constant concentration called X_n). The growth rate of the cell is a function of the rate of this conversion. The rate of conversion is a function of the individual rates of catalysis of the catalysts that comprise the cell. The individual rates are proportional to concentration of the catalyst concerned and to the interaction energy between the particular catalyst and transition state. The interaction energy is a function of the relative shapes of the interacting species (the degree of complementarity between them).

There is here a hierarchy of causation: the top level observable, growth rate, depending ultimately on a low level observable, the complementarity of proto-enzyme and transition state. It seems natural, when developing an expression of the model for computer, to follow the causal chain upwards so that each layer in this hierarchy, as it is reached, rests upon a layer that has already been specified.

The model also makes statements about genomes and genes, but in certain respects these statements are arguably lip-service only, although the software will accord this circumlocution due respect. Because the model does not explicitly incorporate the notion of protein folding the structure of the gene and the structure of the protein are isomorphic. Similarly the number of

copies of each gene is isomorphic with the relative concentrations of the protein specified by that gene (because the relative molecular weights have been thrown into the kinetic constants). The genome is simply the set of genes (= proteins) comprising the cell and thus is isomorphic with the cell. However, the operations on genes and enzymes are logically distinct. Genes may be mutated, duplicated or deleted but do not have active sites or catalytic coefficients. These distinctions of the "real world" are preserved in the model.

All of these objects and operations can be expressed simply and cleanly in Ada. However, additional operations, having nothing to do with the biology or causal hierarchy of the system, need to be specified for the sake of the observer who wishes to see evolution in action.

The Package Coefficients

In FORTRAN, Modula-2, or Pascal all real numbers (numbers with a fractional part) have to represented by use of a predefined type. All three languages provide a type called REAL, and FORTRAN also allows DOUBLE PRECISION (and sometimes higher precision) reals to be declared. As mentioned earlier, because all floating point values are of the same type, there is no way to specify that a particular value, representing a catalytic coefficient for example, is of a distinct kind from some other value representing, say, the length of a cell cycle. (In Modula-2 this can be done, but only by hiding from the program that the value is a floating point number and explicitly providing functions to make available the arithmetic operations associated with them). Ada provides a predefined real type called FLOAT but also allows the programmer to create his/her own floating (or fixed) point types, and values of these types are treated as logically distinct.

Different implementations of high level languages have different underlying representations of the floating point numbers: although there is now an accepted standard defined by the Institute of Electrical and Electronic Engineers (IEEE), it is by no means universally adopted. Even when it is, different compiler writers may choose to use 32 or 64 bits to represent the type REAL (in FORTRAN usually REAL is 32 bits and DOUBLE PRECISION is 64) which means that REAL values may be represented to either 6 or 15 significant decimal digits. With Pascal and Modula-2 if your compiler happens to give only 6 significant figures for REAL values that is simply unfortunate.

Ada allows (indeed, requires) the decimal precision of user defined floating point types to be specified by the user (up to a maximum, System.Max_Digits, that varies from one compiler to another). The current release of the DG/Rolm Ada compiler has a value of 15 for System.Max_Digits (corresponding to a 64 bit IEEE representation). The initial development of the present simulation was done using only 6 significant figures (the simulation runs twice as fast and requires considerably less space) but once development was fairly complete, a one-line change in the package about to be described specified that the highest precision be used.

The requirement for numerical values in the current simulation is that all values have a suitable precision and that a random number generator be available. I decided to provide a *base type* with a random number generator from which all other real numeric types in the simulation could be *derived*. Type derivation in Ada allows new types to be derived from previously defined ones with all the properties and operations of the parent type inherited. Such a derived type is distinct from the parent however: one cannot compare values of the parent and derived type, or assign a value of one to a variable of the other (except by an explicit type conversion).

This base type is exported by the following package:

```
package Coefficients is

   type KINETIC_PARAMETER is digits 15;
   function Random (With_Max_Value : KINETIC_PARAMETER := 1.0)
      return KINETIC_PARAMETER;

end Coefficients;
```

It has to be admitted that the range of numerical programs available in FORTRAN and Algol 68 in NAG libraries are not yet available for Ada. The National Algorithm Group are, I believe, going to produce such a library, and a fairly extensive set of primitives is available in a library defined by the National Physical Laboratory (Symm *et al.*, 1984; Symm & Kok, 1985). Commercial packages, like that of Protran have very recently been released (Spring 1986) in Ada versions. In 1984 very little was available. Fortunately, the fairly rudimentary math library that comes with the Data General/Rolm system proved to adequate for the task in hand, with one major shortcoming. The random number generator provided was of unknown quality, could not be seeded, and always generated the same sequence of values so that successive runs of any simulation using it would have received identical sequences of numbers.

A random integer generator, published by Sedgewick (1984) in Pascal, was adapted for use in the simulation (with an initial seed calculated from the time and date) and, as an interim measure until a good floating point generator could be found, I filled an array of 10000 elements with the first 10000 values produced by the DG/Rolm random float generator so that each call of Coefficients.Random returned a 'random' element in the array by indexing it with a call of the integer generator. Fortunately, I discovered that workers at the National Physical Laboratory had published (Wichmann & Meijerink, 1984) a complete implementation of a generic random float package, which accepts a seed, is default seeded in the same kind of way as I had chosen to initialise the integer generator (using the time) and that generates a uniformly distributed sequence of values. The implementation of the integer and floating

point random number generators can be found in the Appendix. All of the experiments described in the next chapter were run using the Wichmann & Meijerink generator.

The Package Catalysis

The lowest level in the hierarchy of causal effects in the theory is that involving the "box-brick" model of catalysis. This model defines the nature of substrates (transition states), protoenzyme active sites, the interaction energies between them, and the catalytic coefficients arising from their interaction. The model can reliably be expressed in a programming language allowing the specification of abstract data types. The mapping of the model onto an Ada representation is fairly straightforward, and is presented in the package, Catalysis. This package exports the transition states, enzyme binding sites and catalytic coefficients and the operations defining how they are related.

```
with Coefficients;
use  Coefficients;
package Catalysis is  -- an implementation of the "Box-Brick" model

   -- Catalytic coefficients of protoenzymes may have any
   -- value from zero upwards.
   type CATALYTIC_COEFFICIENT is new KINETIC_PARAMETER
      range 0.0 .. KINETIC_PARAMETER'Safe_Large;

   -- The above declaration implicitly declares, in addition to
   -- the usual operators applying to floating point types:
--  function Random (With_Max_Value : CATALYTIC_COEFFICIENT)
--        return CATALYTIC_COEFFICIENT;

   -- Default values are set "randomly"
   type METABOLITE is limited private;   -- transition states

   -- Default value is Universal_Catalyst (see below)
   type ACTIVE_SITE is private;  -- binding site for transition state

   -- Returns the specificity constant Of_Catalyst binding
   -- site Towards_TS (transition state)
   function kcat_Over_Km (Of_Catalyst : ACTIVE_SITE;
                          Towards_TS  : METABOLITE)
      return CATALYTIC_COEFFICIENT;
```

```
    -- Returns a poor but universal catalyst (a "box"
    -- larger than any possible "brick")
    function Universal_Catalyst return ACTIVE_SITE;

    -- Returns a binding site that has maximum complementarity For_TS
    function Perfect_Site (For_TS : METABOLITE) return ACTIVE_SITE;

    -- Returns an arbitrary active site
    function Arbitrary return ACTIVE_SITE;

    -- Returns a catalyst with one of its three dimensions
    -- altered from that passed as a parameter.
    -- This operation is a necessary concession to the
    -- processes of folding and coding not explicitly
    -- represented in the model of the cell.
    function Alter (Catalyst : ACTIVE_SITE) return ACTIVE_SITE;

    -- I/O subprograms declared here allow catalysts and metabolites
    -- to be written to and read from text files.

-------------------------------------------------------------------
private  -- Everything beyond this point is invisible to any program
         -- using the package
-------------------------------------------------------------------

   type DIMENSION is new KINETIC_PARAMETER range 0.0 .. 10.0;

   subtype BRICK_DIMENSION is DIMENSION range
         DIMENSION'First .. DIMENSION'Last/4.0;
   subtype BOX_DIMENSION is DIMENSION;

   type METABOLITE is
      record
         X, Y, Z : BRICK_DIMENSION :=
             Random (With_Max_Value=> BRICK_DIMENSION'Last);
      end record;

   Universal_Dimension :
      constant BOX_DIMENSION := BRICK_DIMENSION'Last * 2.0;

   type ACTIVE_SITE is
      record
         X, Y, Z : BOX_DIMENSION := Universal_Dimension;
      end record;

end Catalysis;
```

The higher levels of the causal hierarchy in the simulation therefore have access to a model of catalysis that is enforced by the software. The only

defined operations for a metabolite (aside from reading and writing their values to text files) involve their interaction with protoenzyme active sites. A function, kcat_Over_Km (k_{cat}/K_M), returns the catalytic coefficient of a given enzyme for a given metabolite, and a second function, Perfect_Site, returns the "perfectly evolved" enzyme for a given metabolite (the box-brick theory supports the concept of the perfect catalyst). The private type construct prevents any program using the package to access the representation of metabolites or active sites directly.

Both active sites and metabolites have default values, and do not require to be explicitly initialised by the user program if the defaults are acceptable. Indeed, METABOLITE is a limited private type and so cannot be given a value in an assignment statement (provision is made to read a metabolite value from a file, however, so that successive simulations may operate upon the same set of metabolites, and so that pathways may be "engineered" to illustrate particular implications of the model). In a model of catalysis taking, for example, electronegativity, ionic and covalent bonding, and inductive effects into account, a universal catalyst may not, of course, be physically realizable.

Because the structure of the active site is not visible outside, it is not possible for the individual $\langle x,y,z \rangle$ dimensions of the "box" to be altered from outside. There is a requirement for such alteration, however, to capture the notion of mutation. For this reason, the function Alter is provided for access by the next package, which will be concerned with the genetic aspects of the model of the early cell presented in chapters 2 and 3. The alternative possibility would have been to export the gene from the catalysis package, and

not the active site. The gene does not form part of the box-brick model however, and would sit unnaturally in a package concerned with enzyme kinetics. For this reason the gene was included in a higher level package concerned with the genome, gene duplication and deletion and mutation. This package will be considered next.

It is a "problem" with abstract data types that the ommission of some operation may render the type unusable. One has to give considerable attention to the properties of the objects desired. In top-down programming, or using the object oriented design methodology of Booch (1983), the operations are identified as a by-product of the design technique. For this reason the consequences of design errors in these techniques can be very serious. Correcting an early error may require extensive reworking of the whole design, and recompilation of many units.

The bottom-up method, however, gives no method for identifying operations, but is extremely forgiving of ommissions. Extra operations can usually just be directly added, with no "design decisions" prejudiced. For the present work, this was an enormous advantage. The initial operations were identified by simply asking: what are the properties of genomes and what should I be able to do with them? This has the consequence that operations are provided which are never actually used. As most of them are simple to implement, the amount of unecessary work done, and code generated, is probably not great. Bottom-up implementation is in this respect not really a design method at all.

The Package Genetics

Each cell possesses a genome that has a certain size (the number of genes) and complexity (the number of unique genes). To implement duplication and deletion it must be possible to arbitrarily select any gene and alter the number of copies of that gene in the genome (deleting it completely if there is initially one copy and it is selected for deletion) or mutate one of the copies. These processes can be hidden from view, occurring as side effects of genome replication. The product of a replication may be a nonviable cell, if a complete block in the pathway to protein has occurred because of the deletion or mutation of a proto-enzyme solely contributing to an essential activity.

The kind of information that we require to recover about genomes as the simulation progresses, apart from their size and complexity, mainly concerns the proto-enzymes they code for. How many substrates does the product of each of the genes in a genome on the average display activity towards? What is the average activity? What is the replication time of a given genome? How many genes in a given genome have products with no catalytic activity at all? The package, Genetics, provides access to this kind of information about genomes. The individual gene is not accessible, however. The specification of Genetics is somewhat larger than that of Catalysis because most of the information required from the simulation is at this level of the model. The specification given here was not reached quickly, but only after several complete implementations of the model were built and run.

```
with Coefficients, Catalysis, Text_IO;
use  Coefficients, Catalysis, Text_IO;
package Genetics is

   -- The range of possible genome sizes
   type GENE_NUMBER is range 1 .. 500;

   -- The range of possible genome complexities
   subtype LOCI is GENE_NUMBER range 1 .. 40;

   Default_Size : GENE_NUMBER := 1;
   -- By default, a genome consists of a single gene specifying a
   -- universal catalyst.  Arbitrary or high specificity genomes can
   -- be obtained, however (see below)
   type GENOME (Size : GENE_NUMBER := Default_Size) is private;

   -- Returns whether the genome supports a viable cell
   function Viable (G : GENOME) return BOOLEAN;

   type CYCLE_TIME is new KINETIC_PARAMETER
           range 0.0 .. KINETIC_PARAMETER'Safe_Large;

   -- Returns the time required for a genome to duplicate itself
   -- (using material from the catalysis of the pathway)
   function Doubling_Time (Catalysed_By : GENOME) return CYCLE_TIME;

   -- Returns the number of distinct loci in G
   function Complexity_Of (G : GENOME) return LOCI;

   -- Returns the number of loci in G that specify
   -- protoenzymes with no activity
   function Null_Products_Of (G : GENOME) return NATURAL;

   -- Returns the average number of steps in the pathway to
   -- protein towards which non-null products of G have activity
   function Active_Steps_Per_Product_Of (G : GENOME) return FLOAT;

   -- Returns the average number of products of G having activity
   -- towards each metabolic step in the pathway to protein
   function Active_Products_Per_Step_Of (G : GENOME) return FLOAT;

   -- Returns the mean catalytic coefficient of non-null steps
   -- catalysed by the individual gene products of G
   function Activity_Per_Step_In_Products_Of (G : GENOME)
      return CATALYTIC_COEFFICIENT;

   -- Returns the mean maximum coefficient per gene product of G
   function Max_Activity_Per_Product_Of (G : GENOME)
      return CATALYTIC_COEFFICIENT;

   -- Returns a replicate of G.  Replication may be imperfect
   function Replicate_Of (G : GENOME) return GENOME;
```

```
      -- Returns that genome producing the most specialised set of gene
      -- products possible under the model for the pathway to protein
      function Highest_Specificity_Genome return GENOME;

      -- Returns a "random" genome with the size specified.  The
      -- highest complexity genome possible for the size is returned
      function Random (Size_Of : GENE_NUMBER := Default_Size)
         return GENOME;

      -- Returns a gene number in the range 1 .. Max
      function Random (Max : GENE_NUMBER) return GENE_NUMBER;

      -- Input/Output to text files
      procedure Put (File : in FILE_TYPE; Item : in GENOME);
      procedure Put_Metabolic_Pathway (File : in FILE_TYPE);

   private

      type METABOLIC_STEP is range 1 .. 8;
      type CATALYTIC_ACTIVITY is array (METABOLIC_STEP)
         of CATALYTIC_COEFFICIENT;

      type PROTO_ENZYME is
         record
            Groove : ACTIVE_SITE;
            C      : CATALYTIC_ACTIVITY;  -- convenience
         end record;

      -- Returns the Universal_Catalyst in Groove and its activity in C
      function Universal_Enzyme return PROTO_ENZYME;

      type GENE is
         record
            Copy_Number : GENE_NUMBER  := 1;
            Enzyme      : PROTO_ENZYME := Universal_Enzyme;
         end record;

      type GENETIC_MATERIAL is array (LOCI range <>) of GENE;

      type CHROMOSOME (Complexity : LOCI := Default_Size) is
         record
            Genes : GENETIC_MATERIAL(1 .. Complexity);
         end record;

      type GENOME (Size : GENE_NUMBER := Default_Size) is
         record
            Gene_Map : CHROMOSOME;
            Net_C    : CATALYTIC_ACTIVITY;  -- convenience again
         end record;

   end Genetics;
```

The Cell Package

Most of the hierarchy of levels of discourse in the current model have now been specified. Above, and first specified, is the population of cells, below is the genome and the proto-enzymes that it specifies. There remains only the cell itself.

All of the essential primitives for the biological aspects of the simulation have already been defined. The cell really has only one operation: cell division. A cell can be implemented simply as a record containing two fields: a genome (which determines the cell's growth rate) and (to save frequent recalculation) the growth rate itself.

```
type CELL is
   record
      Genotype : GENOME;
      Rate     : CYCLE_TIME;
   end record;

procedure Divide (Parent   : in CELL;
                  Critical : in CYCLE_TIME;
                  Data     : in out STATISTICS);
```

The implementation of the procedure Divide involves creating two copies of the parent cell (by replicating its genome and calculating the appropriate rate of cell division) and storing them in the population for later restoration when they become due for division. If either of the created cells is non-viable, or if the critical cycle time is shorter than the time required for either cell to complete a cell cycle, then that cell is not stored. Should the population be at its maximum size, and either of the daughter cells are to be stored, a number of cells in the population are arbitrarily killed. Counts of the number

of lethal mutations, of cells rejected by truncation selection, and of the number of cells killed because of overpopulation are kept in a record of type STATISTICS, which the package also defines.

This, together with the previously defined packages, constitutes the primitives for constructing the simulations. The main program does little more than loop round counting cell divisions until the user-specified number have occurred, and sampling the population at the specified frequency. Much of the design effort at this level is simply in deciding the format of the program output, and the number of files to be written to. These decisions will not be discussed here.

Chapter 6

Experimental Results

Preamble

Given a particular value for the constants *A* and *B* in the Lennard-Jones equation, and a particular metabolic map, there is a simple way of looking at the evolution of the population of cells described in chapters 2 and 3 and brought to life, *in numero*, in the previous chapter: it is a search. The search explores two domains: protein sequence space (represented in the model by a set of three dimensions), and the genetic composition of cells. The objective is to maximise the frequency of cell division.

The model outlined in the second chapter clearly adopts the view of Maynard Smith (1961) rather than Koch (1972) concerning the location in sequence space of contemporary sequences. That is, for the model to be a reasonable approximation of reality, contemporary sequences must generally have been reached from primitive sequences by a series of sequence changes all of which improved function. Modern sequences, to put it succinctly, have arisen by selection (not excluding drift among functionally similar sequences). Koch assumes that this has not been case, that modern sequences cannot be reached by a series of favourable mutations from their primitive ancestors because there are points where two or more simultaneous changes are required to improve function. Hence Koch's argument that untranslatable intermediates have been important in catalytic evolution.

The refinements to the model introduced in the third chapter do not include gene inactivation and reactivation (inactive gene products are allowed, but have an allocation cost), though such a change could readily be accommodated.

The search is "random" with respect to function, in that the variations which occur are not dictated by functional considerations. In another, vitally important, sense they are *not* "random", for the *corresponding continuity of form and function* (Cairns-Smith, 1971) of protein sequences ensures that the results of local changes have only slight global consequences. Whenever a dynamic continuous process intervenes between "genotype" and "phenotype" this is the case, a point that Conrad (1974, 1983, 1985) has discussed. Under the present model, this continuous dynamic process is represented as two sets of equations. The first is the Lennard-Jones function, which takes the genetic specification of a proto-enzyme (its axes) and yields a set of catalytic coefficients; and the second is the expression for the flux, derived from the simple Michaelis-Menten equations for the eight unimolecular first-order reactions of the cells' metabolic map, which takes the catalytic coefficients for each of the proto-enzymes of the cell, combines them, and yields the growth rate of the cell. The effects of single changes are constrained by limiting the effects of any genetic event to a single copy of a single gene, and the effects of mutation to a single component of any sequence (one of $\langle x,y,z \rangle$ for some proto-enzyme active site). Each mutational change is also restricted in magnitude to fifty per cent or less of the existing value.

The evolutionary search being modelled is clearly Darwinian, being driven by natural selection. Natural selection is the gravitational force of the search space (except in the analogy the landscape is "upside-down" and peaks are perceived as troughs). One often thinks of gravitation as being a global rather a local effector but this is not so from the viewpoint of any object being acted upon. The local topology is also important, which is why a

rucksack carelessly dropped on the slopes of Ben Lomond rarely ends up in the Loch. In the protein sequence search space, a sequence is accessible if it can be reached by a series of single alterations from the initial sequence all of which preserve or improve function, and inaccessible if it cannot.

The simulation, of course, addresses the question, what is the topology of the search space as perceived by the cells in the model? How many peaks are there? Note that this is not simply a question about the activity of the individual protein sequences. It may well be that the highest activity peaks in the protein space are represented by monofunctional species, and that they are accessible from most arbitrary points in the landscape, and yet they do not represent the best sequences for the proteins of the cells. The cells are being selected for growth rate, not, directly, for the activity of the individual enzyme species that comprise them. The number of peaks in the cells' search space may be very different from the activity landscapes of the individual enzymes. Not only this, but it may be the case that sequences which are not accessible by the criterion of the previous paragraph can be selectively reached by the population, because functions may be exchanged given appropriate genetic backgrounds.

There is one respect in which the current simulation does not represent a *typical* evolutionary problem. The topology of the search space is not shifting. According to the Stationary hypothesis of Stenseth & Maynard Smith (1984) evolutionary change is driven by (abiotic) environmental change. In the absence of such change, evolution ceases. (On this hypothesis, environmental constancy explains the long periods of stasis that so impressed

Gould & Eldredge [1972, 1977] and Stanley [1975, 1979]). The environment in the current model is constant. The cells populating that environment initially have no evolutionary history (they have arisen *de novo*) and are not optimally adapted to it. They therefore occupy an unstable position on the adaptive landscape. The population explores the space until it finds a local peak. Evolution will therefore cease at some point. Whatever the merits in natural ecosystems of the Red Queen hypothesis of Van Valen (1973), according to which a population has to run all the time to stay in the same place and evolution never ceases, in the world of the model the Stationary hypothesis holds.

In chapter 2 we saw that, where the space does not represent sequence changes, but only copy numbers, there are many local peaks, and if only single duplications and deletions are allowed a population will often come to rest on a peak some distance from the optimum. What happens when mutational (sequence-changing) events are allowed? Does the topology change so that there is now only one peak? This seems to be the assertion of Kacser & Beeby (1984) when arguing that modern cells with high-activity monofunctional enzymes will inevitably arise.

The simulations will look at a only a small number of landscapes. The first set will be characterised by values of A and B, and substrates, which allow fairly high activity in multifunctional enzymes. The second set will be closer the world-view adopted by Kacser & Beeby (1984), in that the price of multifunctionality on activity will be high.

Structure of the Experiments

There a number of parameters that can be passed to the simulation program described in the previous chapter. These parameters will be described here.

The user will generally explicitly specify the following:

i) The size of the experiment (in cell divisions);

ii) The frequency of sampling (in cell divisions);

iii) The truncation factor (see below);

iv) The frequency (in samples) that individual genomes are printed.

Additionally, the simulation program searches the local filespace for three input files. The first of these is a text file, which may have been produced by a previous execution of the program, or by the user (using a text editor). This file specifies whether the pathway is a straight or branched chain, and gives the structures of the eight metabolites (their $\langle x,y,z \rangle$ dimensions) that form the pathway. The second file, which is not a text file and so can only conveniently be produced by a program, is a list of one hundred genomes for the initial population. The third file contains the initial simulation "time", so that resumed experiments have continuous values for time over successive runs. None of these three input files need be present to run the program (default values are provided).

The experiment proceeds in the following way: firstly, the user specified parameters (i-iv above) are read. Then an initial population of one hundred cells is generated. The program checks to see whether these cells are to be

read from an input file (to resume an earlier experiment). If this is not the case, the cells of the initial population are identical by default, each cell having a genome with a single gene specifying a "universal catalyst". This has one important consequence: in the initial cells there are no possible favourable gene duplications. Duplicating all of the genes in a genome has no effect on the duration of the cell cycle and is hence selectively neutral under the model. Mutational improvement is also impossible (in some of the earlier experiments some initial "fine-tuning" could be done but the later experiments start with the "best possible" universal catalyst and all mutations are unfavourable) and so the evolutionary route of the initial population depends at first on neutral duplications occurring. Once it has been formed, the vital statistics of the initial population are reported.

The mean cell cycle duration of the initial population, and each subsequent sample, is used to calculate a "critical period". The duration is multiplied by the truncation factor supplied by the experimenter. Any daughter cell in the next sample interval which has a cell cycle longer than this critical value is killed at birth (truncation selection). A value greater than 1.0 for the truncation factor results in a critical period longer than the mean of the previous sample and so allows the survival of cells which are not as fast as the average cell of the previous sample (weak selection). A value less than 1.0 requires better than average performance for survival (strong selection).

The initial population is permitted to grow (by cell divisions that may yield cells which are genetically different from their parents because gene duplications and mutations may occur) until the maximum size of the population

is reached. The statistics of the expanded founder population are reported, as are the genomes of its fastest and slowest cells, and the experiment 'begins'.

Cell divisions are counted (whether or not either daughter cell survives) and after each sample interval the value of the critical period is updated. Note that sampled cells are drawn from the population, and cells aborted at birth are thus never sampled. For the same reason, neither are cells that are dead at birth (because of a lethal mutation). If the population size is at its maximum, a number of cells (one to five) are killed "randomly" (with no reference to their growth rates) whenever a successful cell division occurs, to make room for the new daughter cell(s). At the end of each sample interval, the statistics of the sample are reported, and, if the genomes of the extreme cells in the sample are to be reported (this is the case when the number of the sample is exactly divisible by the number provided by the user for the "frequency" of such reporting) these are also reported.

If the population survives the duration of the experiment (the total number of cell divisions specified) one hundred of its cells are stored permanently on file, so that the experiment can be resumed at a later time. The cells are stored in machine readable form, but the first and last cells stored are explicitly reported to the experimenter (that is, are written out to a text file).

An Introductory Example

The first set of data to be presented shall be used to illustrate the nomenclature I will use in the figures, and to make some general points. Figure 6.1.1 summarises the progress of the simulation. It comprises five graphs.

The first graph (6.1.1a) shows the size of the population throughout the experiment. The population remains fairly constant at about three and half thousand. The truncation factor in the experiment is 1.08, so selection is very weak and the population is not subject to wide variations in size. However, the experiment shown is not, in fact, a single execution of the simulation program, but ten such executions. Each one of these starts with a population of one hundred cells, drawn, for the second and subsequent runs, from the final population of the previous run. The graph shows that there were initially a number of small runs (two to fifteen thousand cell divisions) followed by a long one (seven hundred and fifty thousand), a short one (five thousand), and three longer runs. The transitions from one run to another force the population through a moderately sized bottleneck.

Figure 6.1.1b shows the increase in average growth rate plotted against the number of cell divisions. The units are arbitrary (units of protein produced per unit time per unit mass or volume) and shall be used in all the data sets. The increase in growth rate was initially very rapid, becoming less steep as the experiment progressed. At the end of the experiment (after approximately 1.5 million cell divisions) the growth rate was still increasing gradually.

Figure 6.1.1c shows the change in the average genome *size* (the total number of genes per genome) and average genome *complexity* (the number of distinct genes in each genome). This graph is much 'noisier' than the one of growth rate. The size and complexity of the genome fluctuates, but there are two clear periods when size increases quite substantially, while the complexity of the genome, after an initial rapid rise, climbs unsteadily upwards throughout most of the experiment. It is not clear whether the reversal of this trend at the end of the experiment would continue if the experiment were to be resumed.

The final two graphs of figure 6.1.1 (d and e), are concerned with the amount of variation present in the sampled population. The variation in growth rate, shown in figure 6.1.1d, fairly high in the expanded initial population, is soon reduced, as expected for characters closely related to fitness (in the model, all of the variation in fitness is due to variation in growth rate). The size and complexity of genomes, on the other hand, are highly variable throughout the experiment, the degree of variation fluctuating considerably.

Variation in the size and complexity of genomes arises, of course, by replicational or segregational 'errors'. The probability of such changes occurring in a genome in these experiments is (somewhat unrealistically) independent of the genome's size. Each cell division produces a parental-type daughter cell and has a fifty per cent chance of the other daughter being a mutant cell, which differs from its parent by a "sequence" alteration (a mutation) in one of its genes, or by a gene duplication or deletion. Duplication, deletion and mutation of a gene are equiprobable events. In this experiment, due to a programming error, the probability of any particular gene

Introductory Experiment;

Duration: 1.53 million cell divisions.

Sampled: Every 500 cell divisions.

Truncation factor: 1.08

Figure 6.1.1

a) Population: reduced to 100 at end of each session

b) Mean growth rate of population

c) Mean genome size & complexity

d) Variation in growth rate

e) Variation in size & complexity

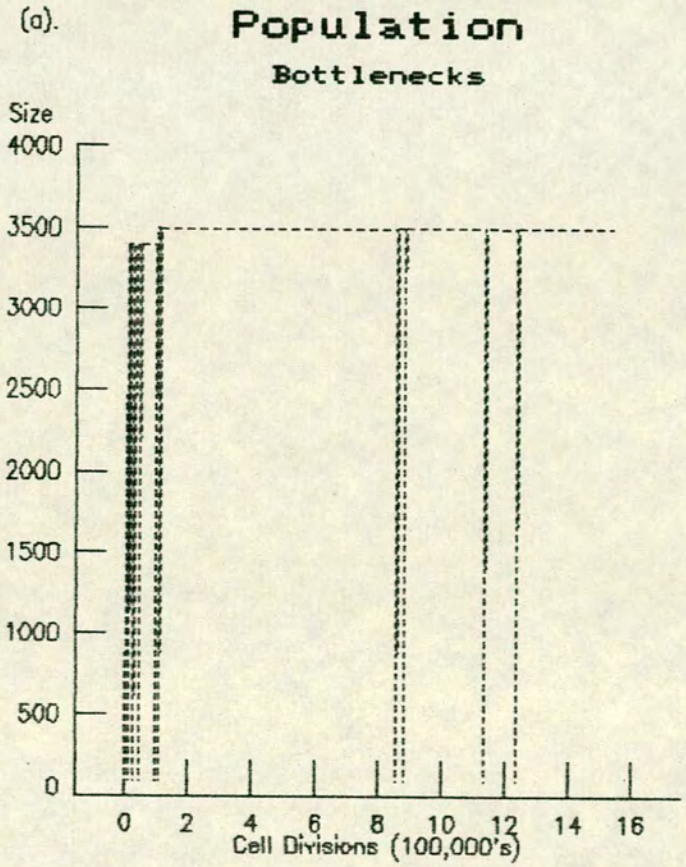
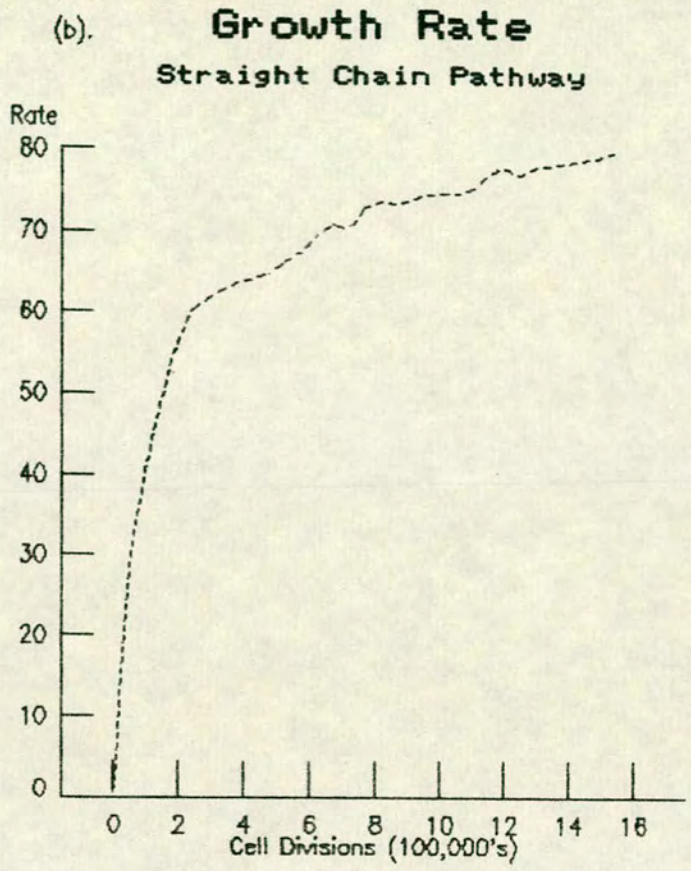


Figure 6.1.1


(b).
Growth Rate
Straight Chain Pathway
Rate
80
70
60
50
40
30
20
10
0
0
2
4
6
8
10
12
14
16
Cell Divisions (100,000's)

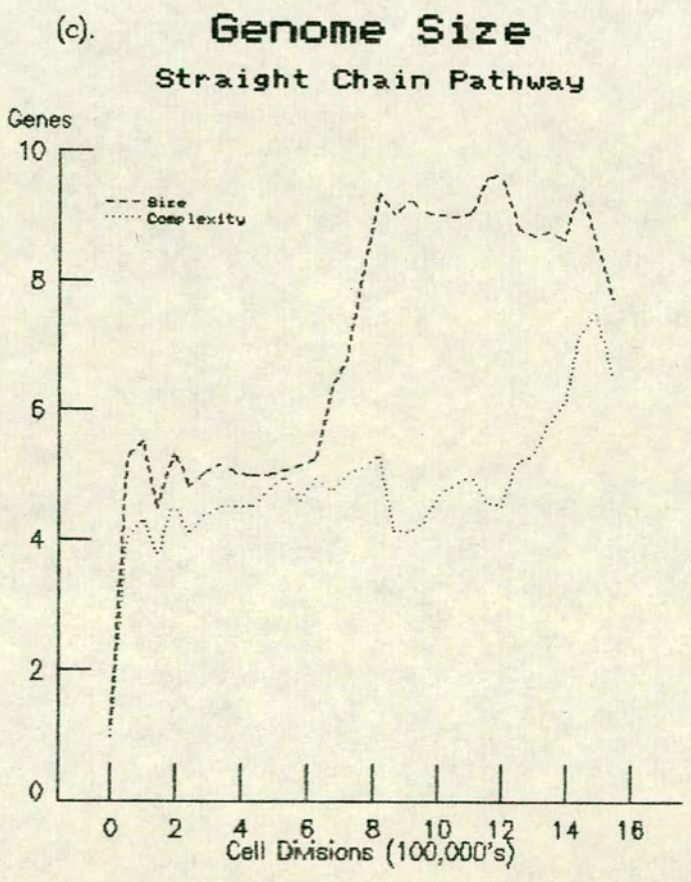
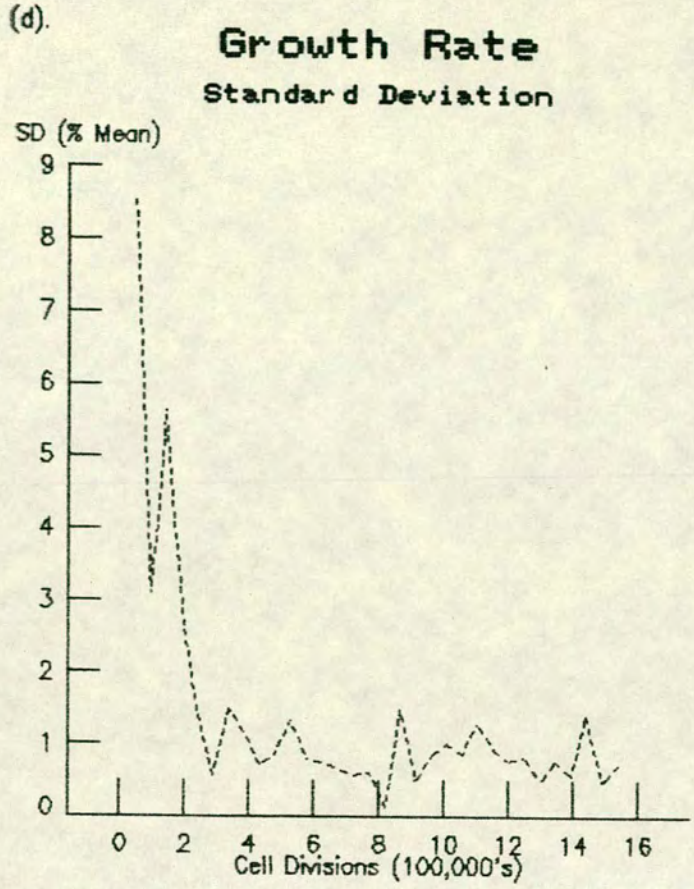

(c).
Genome Size
Straight Chain Pathway
Genes
10
8
6
4
2
0
Size
Complexity
0
2
4
6
8
10
12
14
16
Cell Divisions (100,000's)

Figure 6.1.1

(d).



(e).

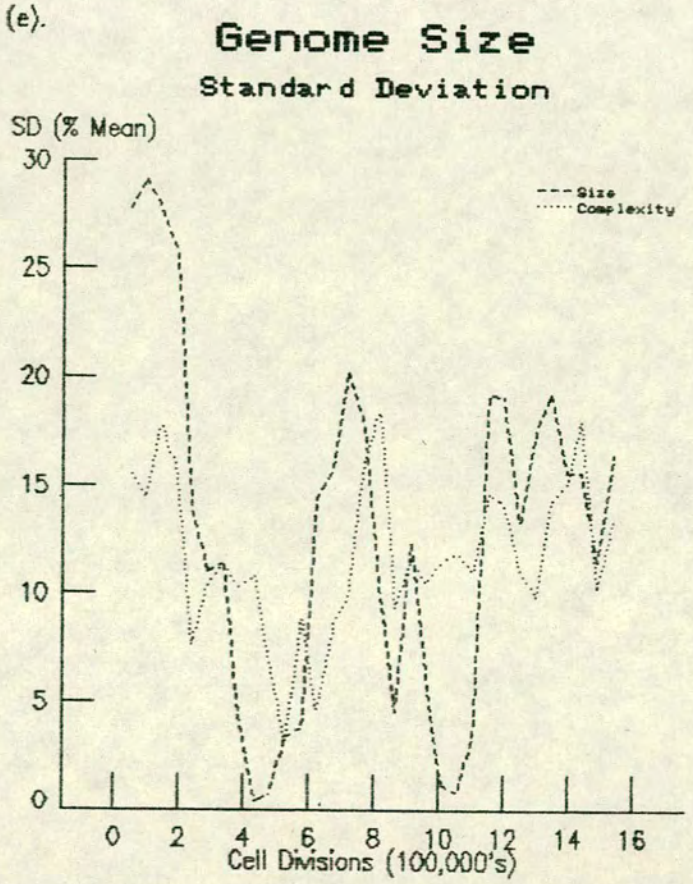


Figure 6.1.1

being affected by mutation, duplication or deletion is independent of how many copies there are of it (though any such change affects only one copy, of course). This is corrected for all later data sets.

The metabolic map of the cells in this experiment are shown in the first chart (a) of figure 6.1.2. This is a stacked histogram showing the three dimensions of each of the transition states for the eight first-order reactions of the pathway. In the top right hand corner of the chart is given the separation (R_{max}) at which binding energy is maximum (by the Lennard-Jones 6-12 function). The larger this value is, relative to the differences in size between the substrates in each dimension, the less the discrimination between the substrates can be. The current value of 0.7647 must be compared with differences in size ranging between zero and two.

Careful examination of the figure will show that several of the molecules have similar values for one or more of their dimensions. For example, the species numbered 1, 3, and 5 have identical *x* dimensions, and numbers 2 and 3 have identical *z* dimensions. This pathway, then, may favour some degree of multifunctionality, since maximum active-site/transition state interaction for the dimensions concerned does not entail discrimination between competing reactions. This may explain the result shown in figure 6.1.2b, which displays the average number of reactions for which each proto-enzyme has some activity. The number drops rapidly from eight for the universal catalyst of the ancestral cell to less than four, but then shows no sign of any further decline and actually rises a little.

Introductory Experiment;

Figure 6.1.2

a) Metabolic pathway – Eight reactions

b) Mean number of reactions catalysed by each enzyme.

c) Extractable enzyme activities from single cells sampled when stated.

(a).

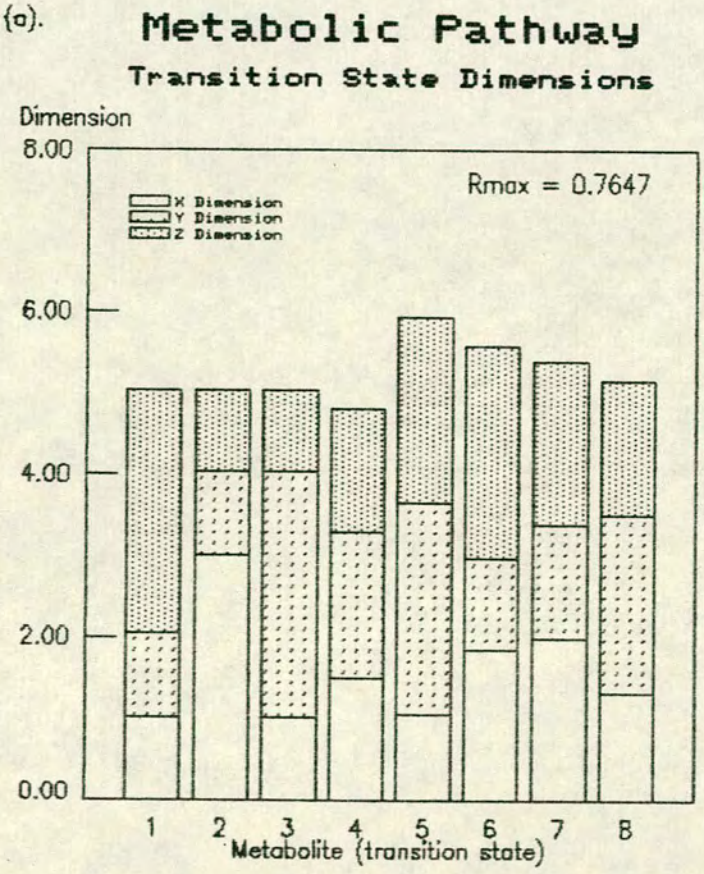
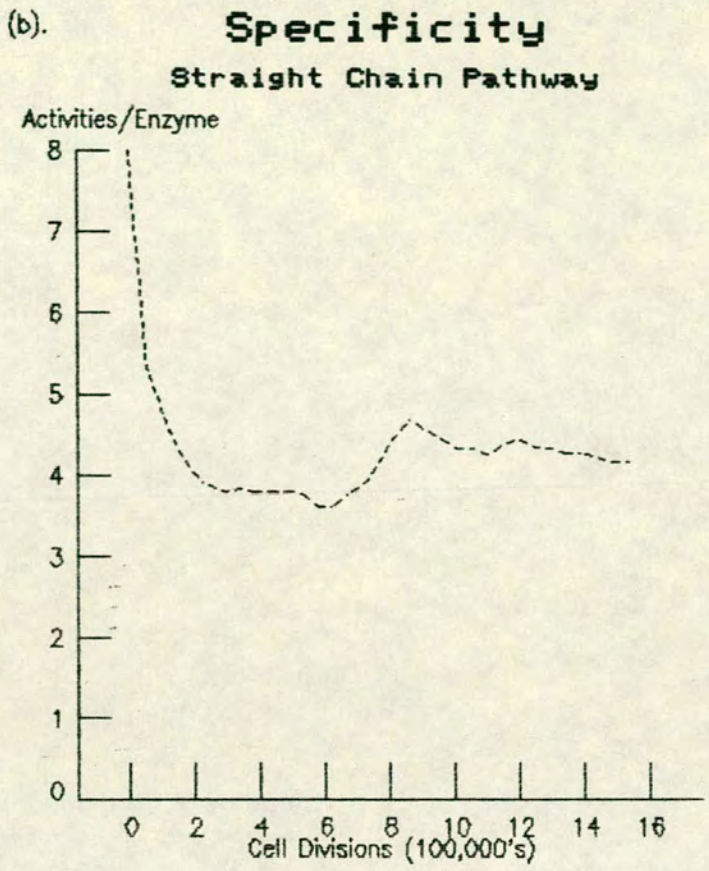


Figure 6.1.2

(b).



(c).

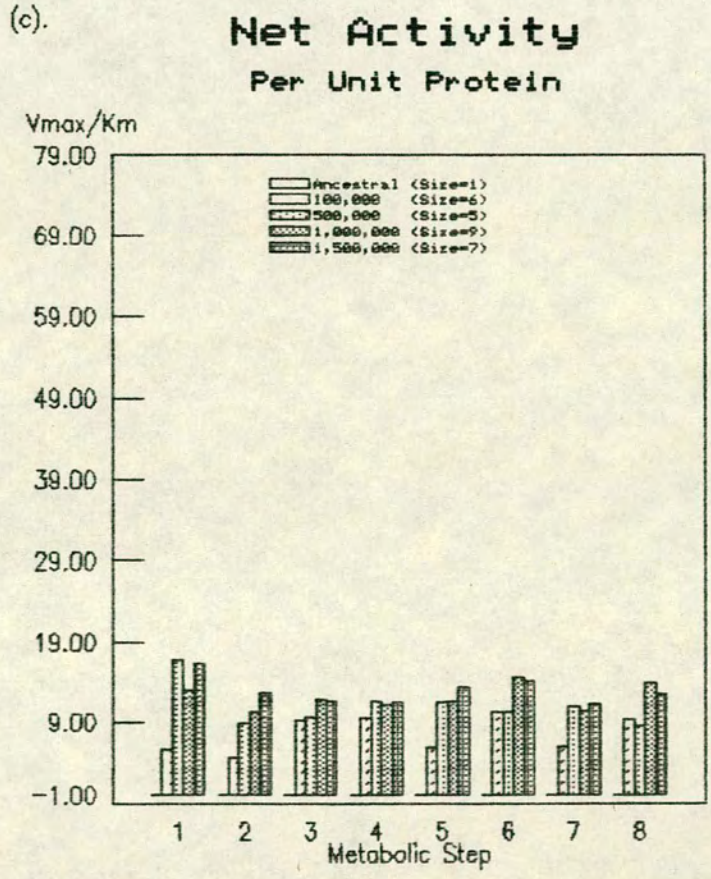


Figure 6.1.2

Figure 6.1.2c shows the net measurable activities (V_{max}/K_M) extractable from individual cells drawn from the population, towards each metabolite after the number of cell divisions stated in the key to the figure. Note that the scale is drawn from -1.00 so that the bars in the chart are above the x-axis and null or very low activities (like that of the ancestral enzyme) are visible. A number of proto-enzymes may contribute to each activity. Note that the chart shows activity per unit of total cell protein.

Surprisingly, only for one of the eight reactions is the pattern of activity one of consistent increase between the sampled cells (metabolic step 3). The details of the individual cells summarised in figure 6.1.2c are given in figure 6.1.3, which comprises six histograms each headed Activity Profile. Figure 6.1.3a shows the distribution of activities in the ancestral cell, which are not distinguishable from null in the scale on figure 6.1.3. The growth rate of the cell is given in the top right hand corner of the chart (G = 0.00574). The largest activities are shown towards the first, second, third and fifth metabolic steps (k_{cat}/K_M slightly greater than 0.001), the smallest ($\simeq$ 0.0006) towards the fourth.

Figure 6.1.3b is necessarily more complicated, for there is now not one proto-enzyme but four to display. Below the growth rate (43.66) of the cell the size (6) and complexity (4) of the cell's genome are given. The key, at the top left of the chart, reveals that there are two copies of gene 1 and gene 2 in the genome and one copy of each of gene 3 and gene 4. The histogram plots k_{cat}/K_M (not V_{max}/K_M) and so, unlike the net activity histogram of figure 6.1.2c, does *not* reflect the relative enzyme concentrations. The heights of the

columns would be the same whatever the number of copies of each of the genes there happened to be. This allows the contributions to activity due to the structure of the gene to be clearly distinguished from the contribution due to the number of copies of it.

The proto-enzymes are ordered, in this and all subsequent histograms of this type, from left to right by increasing size of their first (x) dimension. Enzymes with the same first dimension will generally, but not always, be ordered by increasing y dimension. This ordering is to ensure that similar (for example, recently diverged) sequences are close to each other in the bar chart.

After one hundred thousand cell divisions, the descendant cell shown in figure 6.1.3b has only one proto-enzyme (that coded for by the two copies of gene 4) with activity towards the second metabolic step and one (coded for by the two copies of the gene 1) for the third step. On the other hand, all four proto-enzymes have some activity for the first metabolic step (though these are low for all but one of them). The number of activities displayed by each proto-enzyme coded by the genome of this cell varies from three (for the product of gene 4) to six (for the product of gene 3): the mean is 4.75. The mean value for each of these activities is 13.60.

Four hundred thousand cell divisions later, the growth rate of the cell sampled (figure 6.1.3c) is 66.48, fifty-two per cent higher than the earlier cell. The particular cell depicted has no duplicate genes; both the genome size and complexity are five. Despite the extra proto-enzyme it is still the case that

the second and third metabolic steps are each acted upon by only one proto-enzyme species. The activities have increased considerably for all steps except the eighth, with the mean (non-zero) activity being 23.26. As expected, this increased activity is paid for by increased discrimination: the mean number of steps catalysed by each proto-enzyme of the cell has decreased to 3.80.

A comparison of the two cells supports the view that the evolution of catalytic activities proceeds from broad spectrum low-activity catalysts to narrower-spectrum higher activity catalysts. The increase in each of the activities of the proto-enzymes more than offsets the reduction in their number. An examination of a cell sampled after one million cell divisions, however, shows that this is not always so. This cell, depicted in figure 6.1.3d, has a growth rate of 74.75 and possesses five distinct genes, of which three are single-copy and two are present in three copies each. The average (non-zero) activity for the proto-enzymes specified by the cell's genome is 22.54, less than that of the earlier cell, while the average number of activities displayed by each proto-enzyme, 4.20, is greater. Again there is a trade-off between the number and level of each activity with the trade favouring larger numbers of lower activities.

It is also interesting to note that the distribution of activities for particular proto-enzymes can very closely resemble each other. This is so for the products of the fourth and fifth genes in figure 6.1.3d, which have similar levels of activity towards steps 1, 2, and 6 (gene 4 has significantly higher activity towards step 2).

178 ⇒

Introductory Experiment;

Figure 6.1.3

a) Activity profile of original cell.

b) Activity profile of a cell sampled after 100,000 cell divisions.

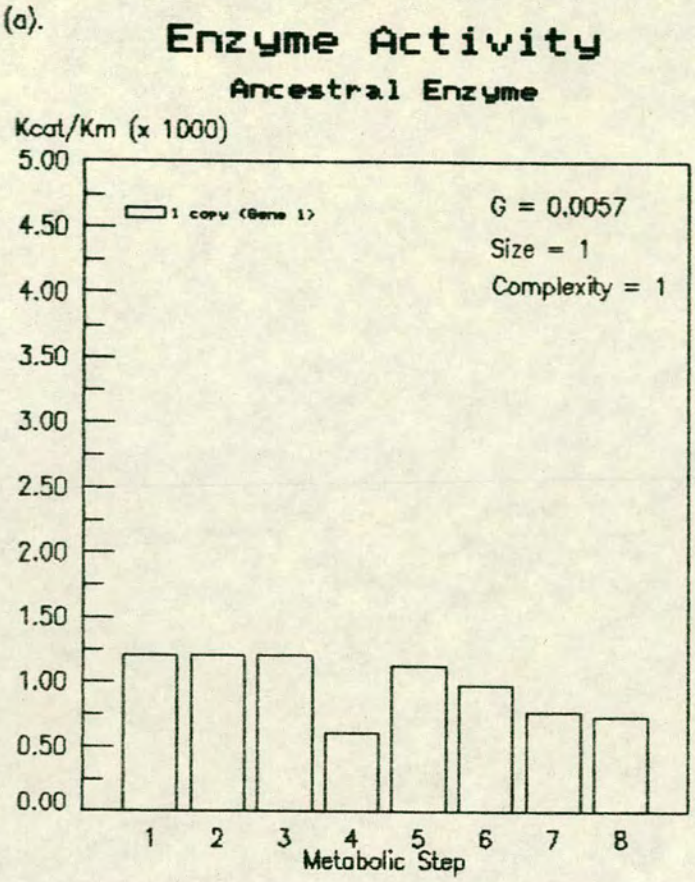
c) Activity profile of a cell sampled after 500,000 cell divisions.

d) Activity profile of a cell sampled after 1,000,000 cell divisions.

e) Activity profile of a cell sampled after 1,500,000 cell divisions. The cell has a genome complexity greater than 5, and for clarity the chart is split into two parts.

Figure 6.1.3

(a).



(b).

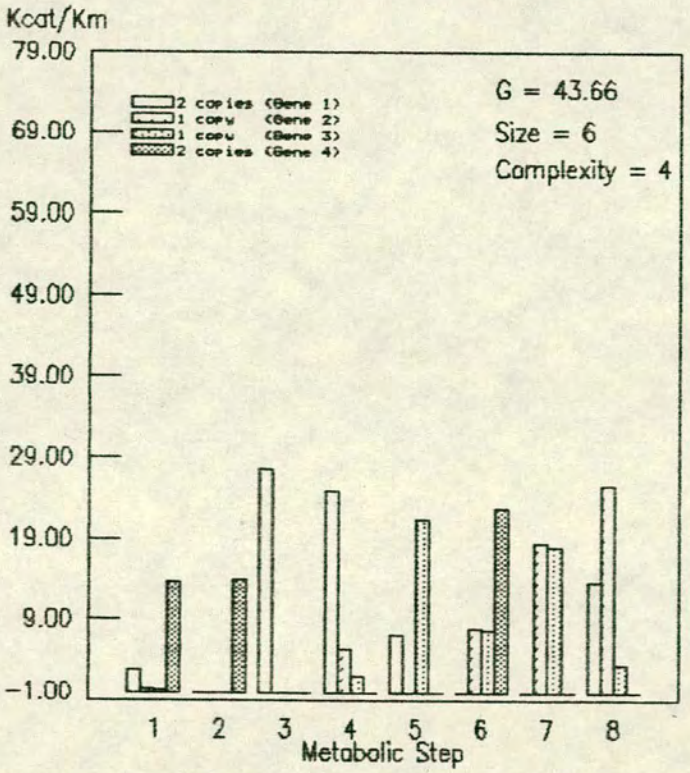
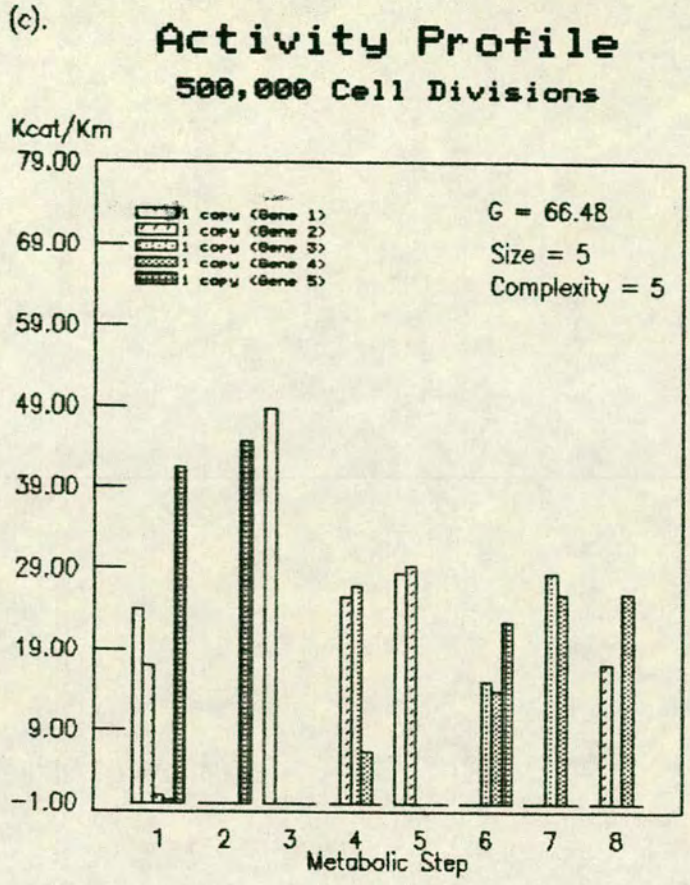


Figure 6.1.3

(c).



(d).

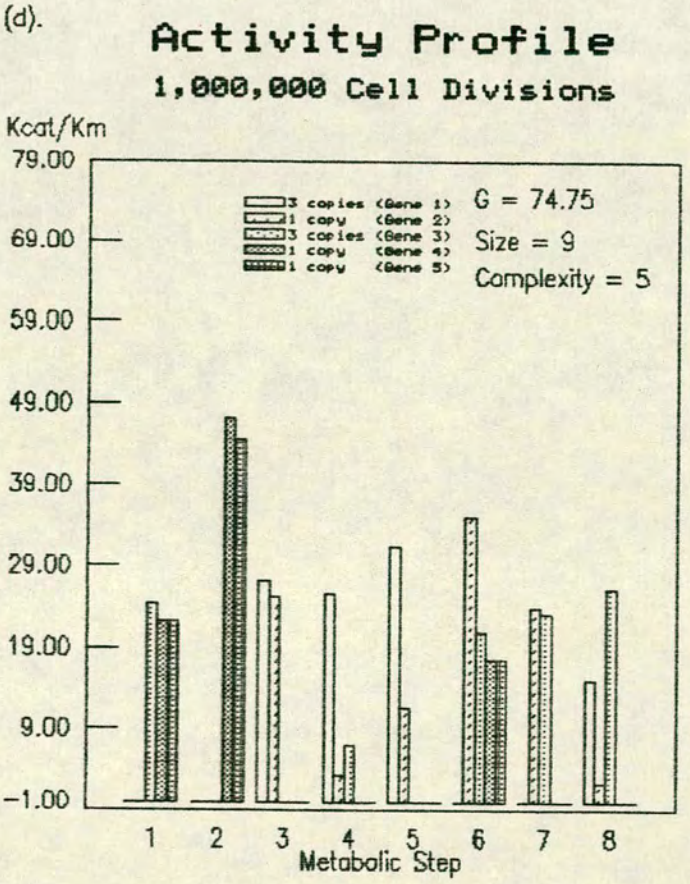
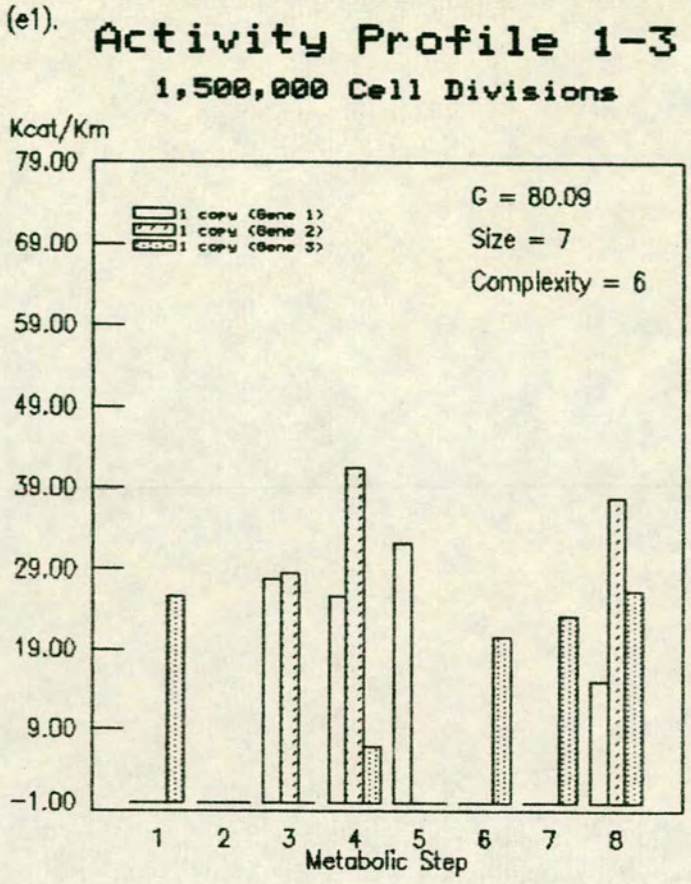


Figure 6.1.3

(e1).



(e2).

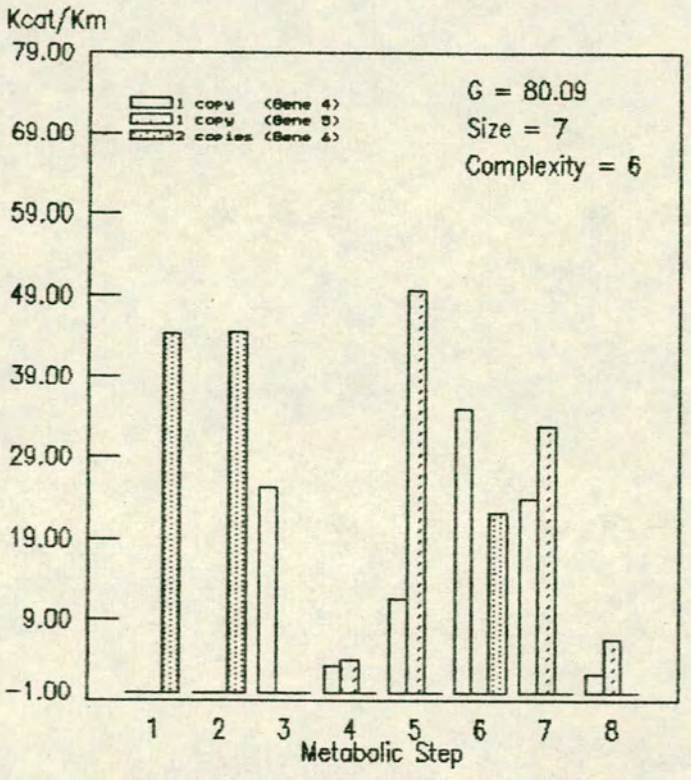


Figure 6.1.3

The cell depicted in the two charts comprising figure 6.1.3e, which was the fastest cell in the sampled population after one and a half million cell divisions, has a genome complexity of 6. Whenever the complexity of a cell's genome exceeds 5, the activity profile of the cell will be split into more than a single bar chart, to retain legibility.

The growth rate of this final cell is 80.09, and one of its six genes is present in two copies. It is clear, comparing figures 6.1.3d and e, that activity has increased, with four of the activities greater than 40.0, compared with only one in the cell shown at one million cell divisions. The mean (non-zero) activity has increased to 24.64, higher than the cell shown at five hundred thousand cell divisions, although the mean number of activities per proto-enzymes remains high (4.17).

Only the product of gene 6 (present in two copies) has activity towards step 2. Step 1, for which all of the proto-enzymes of the first two cells shown had activity, is supported by only two activities in this and the previous (one million division) cell. Step 3, on the other hand, which in the earlier cells was carried by only one proto-enzyme, and at the one million divisions point by two, is now supported by three.

It is interesting to ask what effect the truncation selection has had on direction the population has evolved. A glance at figure 6.1.4 reveals that at the beginning of the experiment, approximately equal numbers of fatalities may be attributed to truncation selection and to the effects of overpopulation, but as the cells develop a history of selection (become adapted) the number of

Introductory Experiment;

Figure 6.1.4

Truncation factor: 1.08

(weak selection).

The graph shows the total number of cell deaths due to

i) Lethal mutations

ii) Truncation selection

iii) Overpopulation (arbitrary kills)

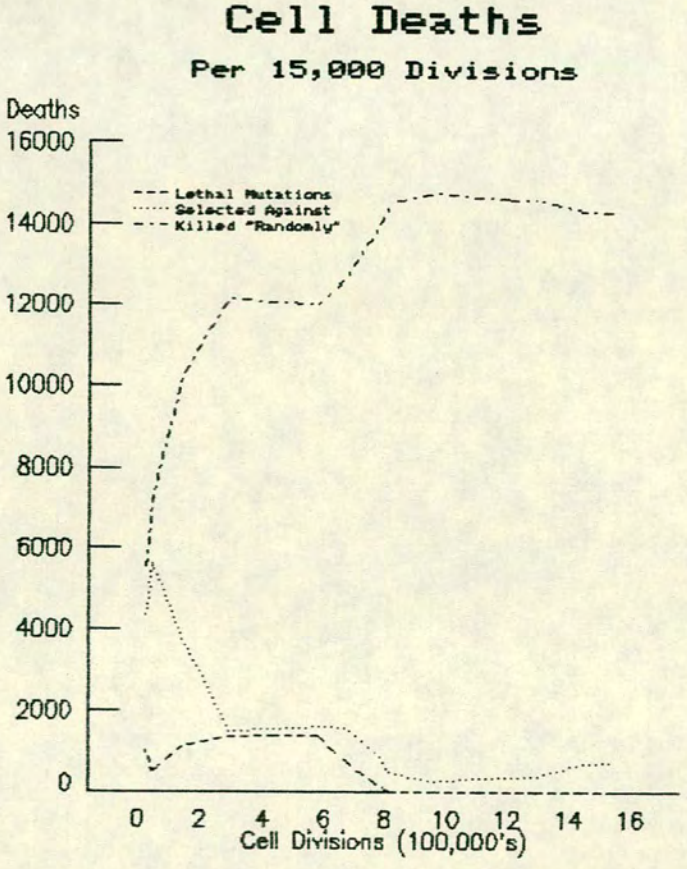


Figure 6.1.4

cells selected against falls very considerably and the vast majority of cell deaths are due to random events having no relationship to fitness. Lethal mutations, interestingly, completely disappear after some eight hundred thousand cell divisions. The differential rates of reproduction of the cells (natural selection) seems to have successfully driven the increase in growth rates without heavy input from the artificially imposed truncation selection scheme.

That the population has not evolved towards monofunctional proteins is no real surprise, in view of the pathway and the values of the Lennard-Jones constants. Some insight can be gained into the evolution of this population by examining figure 6.1.5, which shows the activity profile for a cell with eight monofunctional enzymes. Each of these enzymes has the highest possible activity for the step that it catalyses. The growth rate of this cell is 58.59, considerably less than that of the cell sampled at the end of the current experiment. The expense of producing eight monofunctional proto-enzyme species is not justified for the current pathway, where the similarity of substrates results in multifunctional proto-enzymes giving considerably higher growth rates.

Cell with Monofunctional Catalysts

Figure 6.1.5

Activity profile for a cell with eight maximum–activity monofunctional proto–enzymes.

Although the activity for each catalyst is greater than achieved in the experiment, the return on protein invested is smaller.

Pathway Favouring Multifunctionality - Experiment 1

The first repeat of the introductory experiment that we shall discuss is one of the longest simulations I have run, representing approximately one hundred and fifty seven hours of machine time execution (processor time amounting to about one hundred and forty five hours) on the Data General MV4000 minicomputer used for all of the computational work. The experiment extended to about five and a third million cell divisions (figure 6.2.1a).

The experiment was initiated with a somewhat more stringent truncation factor (1.02) than the previous experiment. Unexpectedly, after some four hundred thousand cell divisions, the mean growth rate levelled off at about 49.8, and then slowly declined over the next eight hundred thousand cell divisions, to a mean of about 49.4 (figure 6.2.1b). The fastest sampled cell at the first-mentioned point had a growth rate of 51.6, compared with 51.0 at the second (after 1.26 million cell divisions), and indeed the range of growth rates of cells in the population also decreased over the interval (48.7-51.6 declining to 48.6-51.0). At this second point, the truncation factor was set to 1.00, and the effect on the growth rate is apparent from the figure (6.2.1b) and from the steep rise in the number of cells killed by selection at that point (figure 6.2.1g).

The experiment followed a quite different pattern from the introductory experiment in terms of the size and complexity of cell genomes (figure 6.2.1c). After one million cell divisions the average genome size was greater than thirty-nine, and the average genome complexity, twenty-one. These figures had

increased to about sixty-one and twenty-nine respectively after one and half million cell divisions. Some thirty fairly extended experiments, run while the program was being developed, had consistently produced genomes with sizes less than ten. Admittedly, several, but not all, of these involved different pathways with metabolite dimensions generated at run-time by a random number generator (the same Lennard-Jones equation and constants applied). Indeed, on the basis of these experiments, the maximum allowed genome complexity for the "final" version of the program had been set to twenty! The higher the maximum allowed value for the size of the genome, the higher the space requirements of the program. However, the first actual repeat of the introductory experiment, not described in detail here because of the effects on that experiment of this limit, generated the pattern of genome size and complexity shown in figure 6.2.1h, revealing that the limit on genome complexity was too low. If the maximum allowed genome complexity is reached, mutations in multiple copy genes, which increase the genome complexity, are prohibited. Presumably as a consequence of this, from the point in figure 6.2.1h where the complexity levels off, growth rate ceased to increase, and actually declined a little (from 70.5 to 69.9). Growth rates can decline, of course, if the truncation factor is greater than one. The factor for that experiment was the same as the initial value for the present one (1.02).

For the current experiment, and all subsequent ones, the maximum allowed genome complexity was set at forty. This complexity was not approached by any sampled cell in this or any subsequent experiment. (Note that genome *sizes* do not create space problems: they are arbitrarily limited to the range 1 to 500).

Experiment 1;

Duration: 5.34 million cell divisions.

Sampled: Every 3000 cell divisions.

Truncation factor: 1.02, then 1.00

Pathway and ancestral cell are
as in the introductory experiment.

Figure 6.2.1

a) Population: reduced to 100 at end
of each session.

b) Mean growth rate of population.

c) Mean genome size & complexity.

d) Mean number of reactions catalysed
by each enzyme.

e) Variation in growth rate.

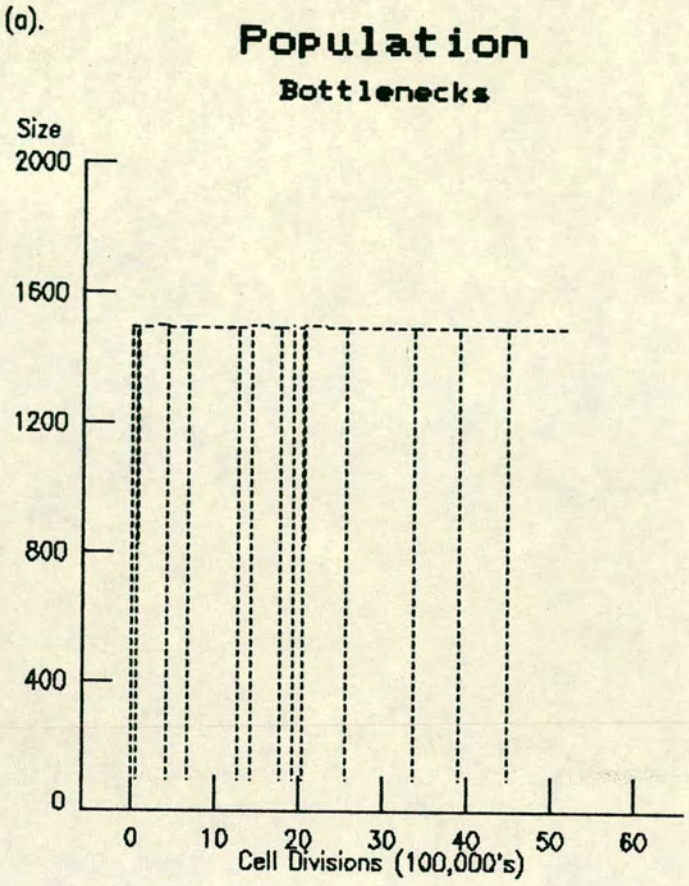
f) Variation in size & complexity.

g) Cell deaths from all causes.

h) Abandoned experiment genome size.

Figure 6.2.1

(a).



(b).

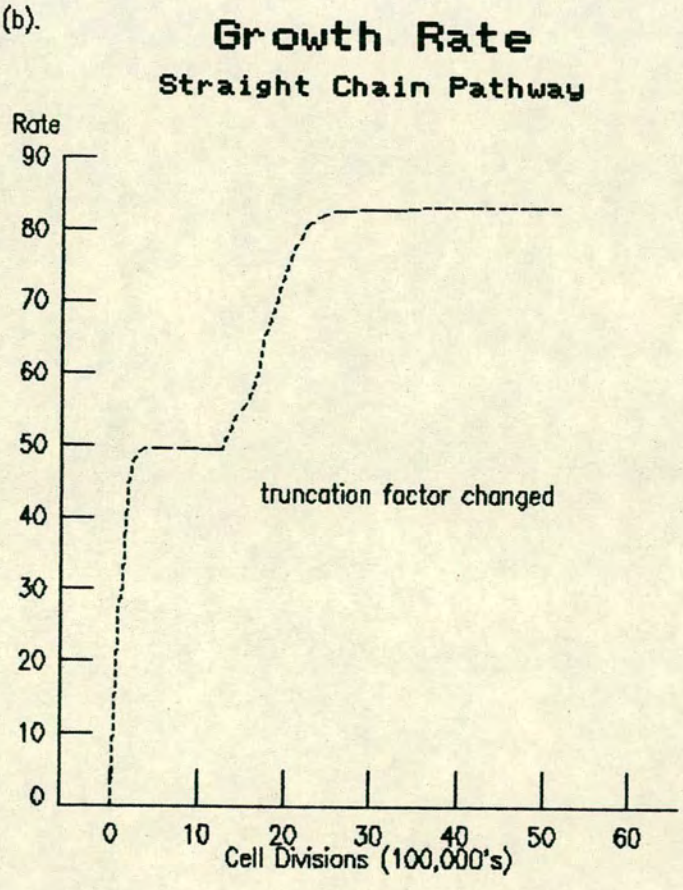
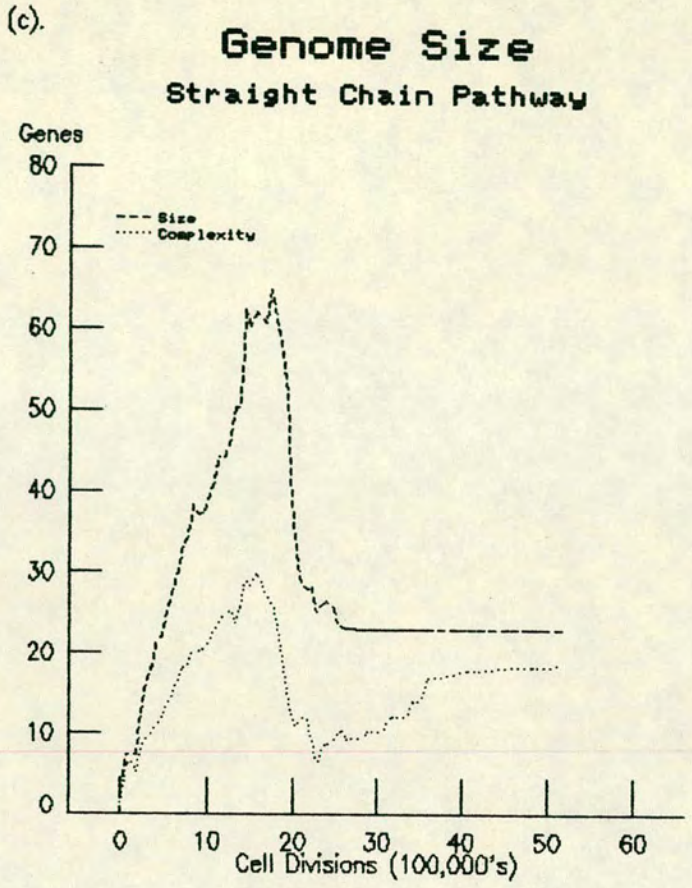


Figure 6.2.1

(c).



(d).

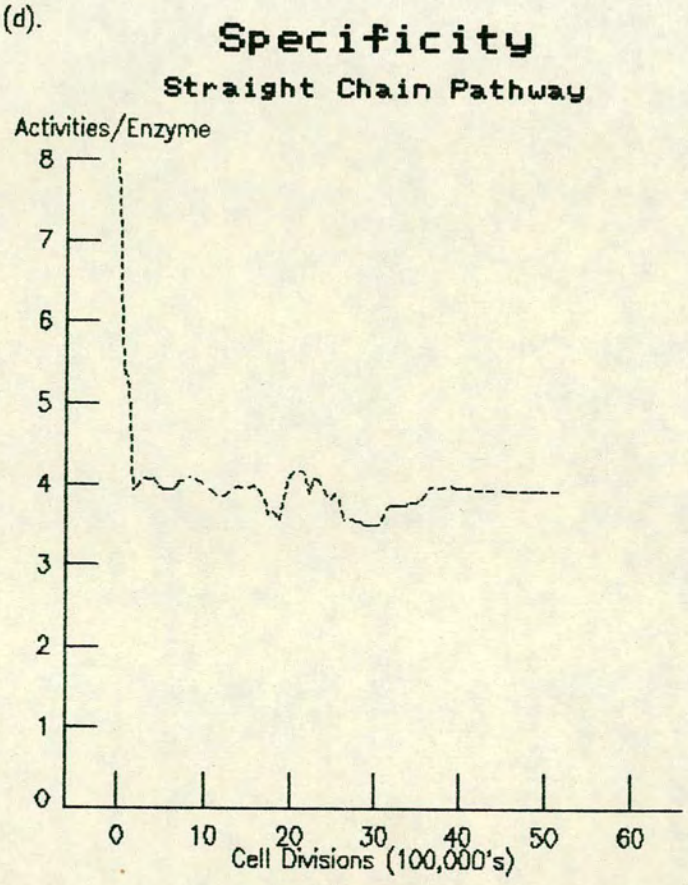
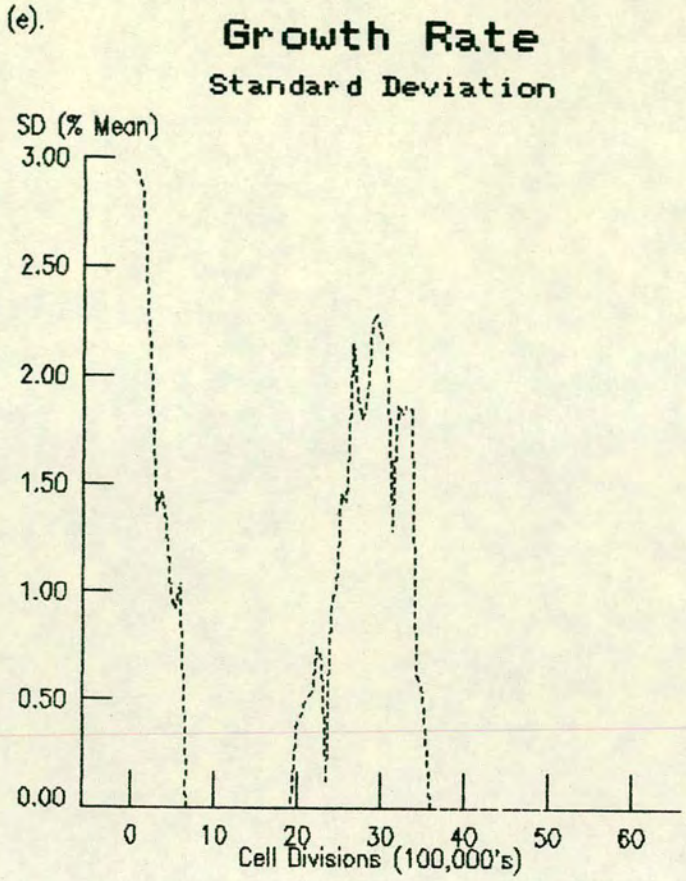


Figure 6.2.1

(e).



(f).

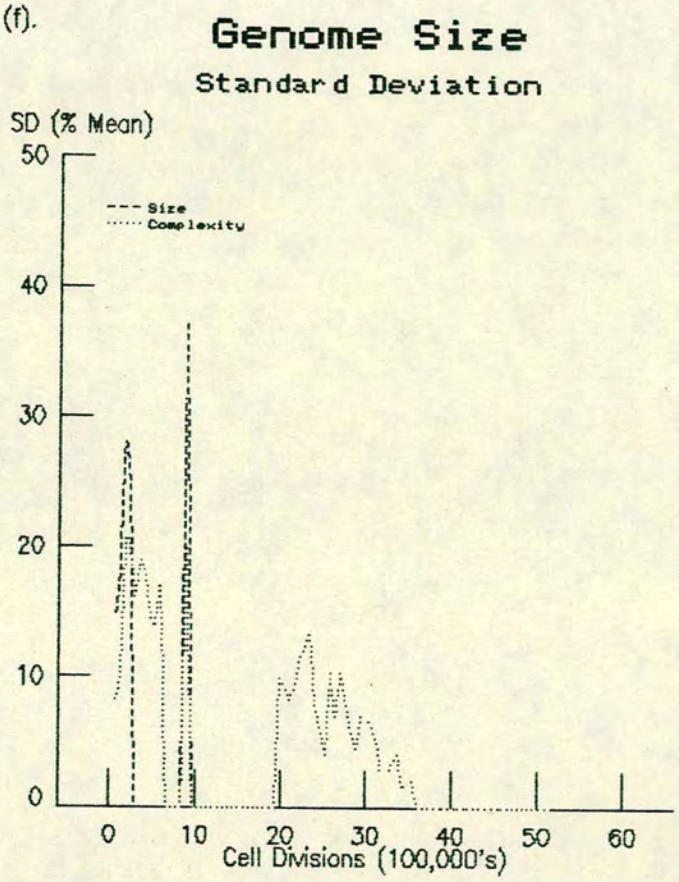
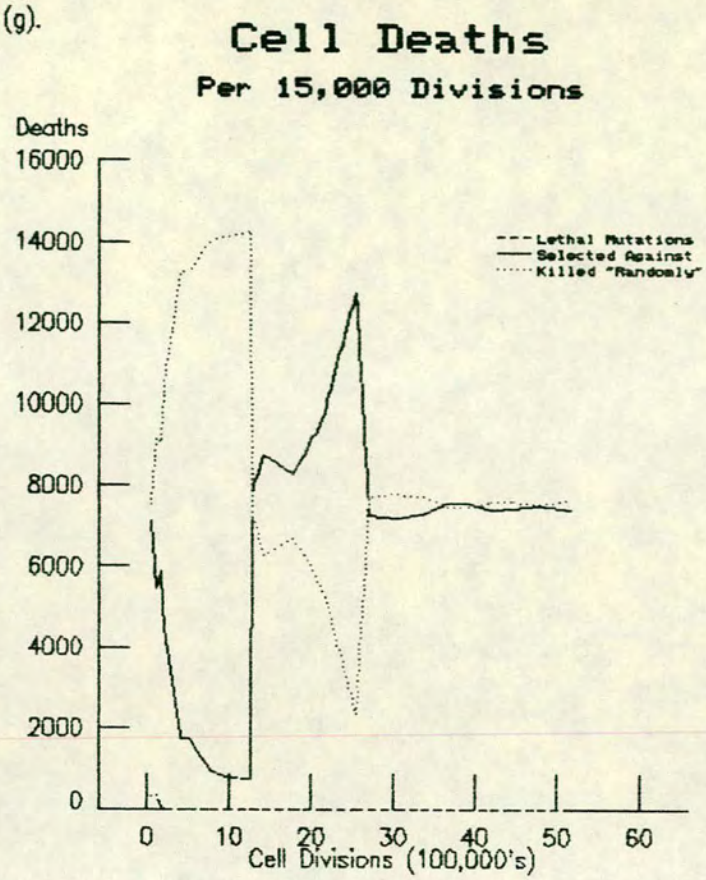


Figure 6.2.1

(g).



(h).

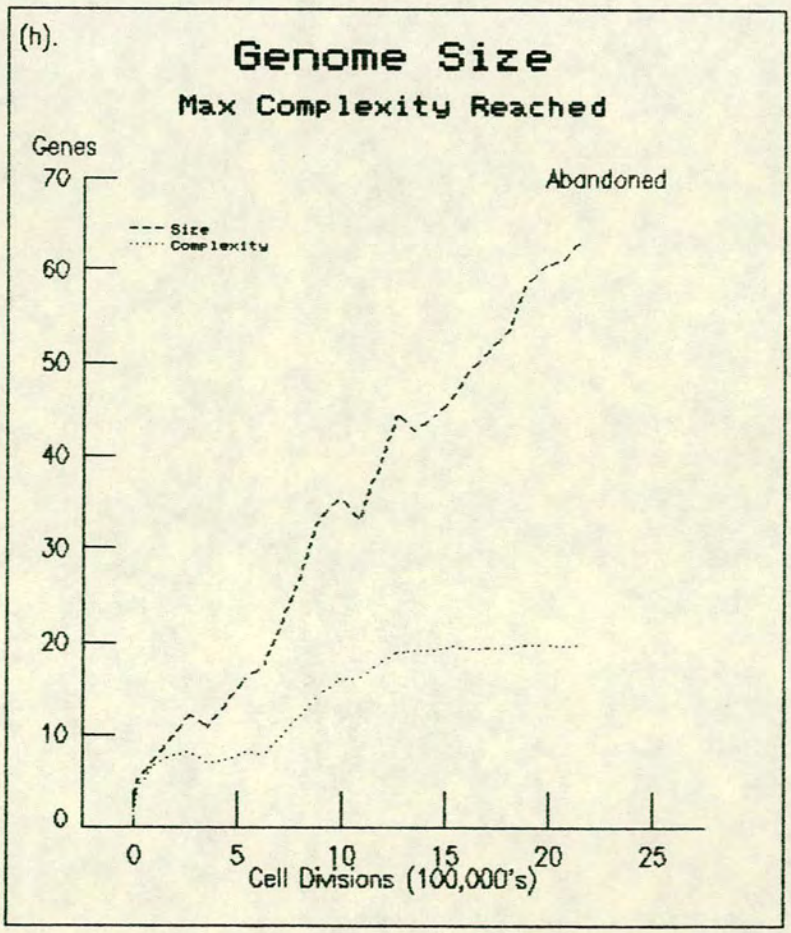


Figure 6.2.1

After the selection intensity was increased, the size and complexity of the genomes of sampled cells continued to rise steeply for three hundred thousand cell divisions. Figure 6.2.1c reveals, however, that the rise then ceased, and after a further two hundred thousand cell divisions, there was a rapid decline in the number of genes per cell. The average size of the genome eventually settled to exactly 23, and all sampled cells from this point had this genome size. There is some evidence of intermittent failure of the package responsible for calculating standard deviations as for long periods of the experiment, all the standard deviations of genome sizes and growth rates were reported to be zero (see figure 6.2.1e,f). Examination of the first such period shows that both mean genome sizes and complexities were changing rapidly, so that the presence of considerable variation is implied. I have been unable to determine the cause of the problem, but suspect that it may lie in one of the Data General mathematics packages used in the calculation of these statistics (and nowhere else in the program). At various points in the experiment, the standard deviation of genome size is reported to be zero (to four significant figures), while that of the complexity is not. This is, I think, a genuine result, and is supported by the fact that at these points in the simulation, all explicitly reported cells had identical genome sizes (but not genome complexities).

As mentioned above, after approximately 2.7 million cell divisions the average genome size remained at 23, and all reported cells in the second half of the experiment had that genome size. This suggests that from this point all gene duplications and deletions were unfavourable. Interestingly, favourable mutations continued to occur and after falling below 10 the average genome

complexity recovered, rising to over 18. Very surprisingly, these mutations did not seem to alter the unavailability of favourable changes in genome size.

To test whether, in fact, this was a genuine phenomenon, or an artifact (it could be that no duplications or deletions were occurring, and so could not be selected for) a number of the explicitly reported cells were examined to see if any potential changes in genome size do increase (or at least maintain) growth rates. The net activities of two of these cells, before and after duplicating/deleting one copy of each of their genes are described in Table 6.2.1. The cells are separated by over one and a half million cell divisions in time, and have quite different genome complexities, but no favourable duplication or deletion is possible in either of them. The same is true of all other investigated cells also. This is one of the truly surprising results of these experiments: that constraints on the genome size can apply so consistently over such a long period, in the presence of considerable variation in the numbers of distinct genes in the genome.

An examination of the table, and the cells described in figure 6.2.2, reveals that many of the 'distinct genes' are, in fact, highly similar to each other. In this sense the genome complexities, which report how many genes with distinct sequences are present, may mislead the reader. The functional similarity of different sequences is very marked, and selective equivalence or near equivalence (neutrality) is clearly present.

Consider in detail the cell sampled after 3.36 million cell divisions (shown in the first half of table 6.2.1) and the cell sampled after 5.0 million cell

Table 6.2.1

SAMPLED CELL AFTER 3360000 CELL DIVISIONS

Genome Size = 23
Genome Complexity = 12
Growth Rate = 82.82979

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	0.00	28.79	38.93	0.00	0.00	0.00	39.77
2	1	0.00	0.00	29.28	38.37	0.00	0.00	0.00	39.87
3	1	0.00	0.00	29.27	38.40	0.00	0.00	0.00	39.86
4	3	0.00	0.00	27.88	38.38	0.00	0.00	0.00	39.80
5	1	0.00	0.00	27.64	40.04	0.00	0.00	0.00	38.57
6	1	0.00	0.00	0.00	4.01	50.33	0.00	31.65	6.44
7	4	0.00	0.00	0.00	3.51	30.62	39.43	26.05	5.52
8	1	0.00	0.00	25.31	3.22	29.15	0.00	32.50	3.10
9	1	0.00	0.00	25.30	3.16	29.39	0.00	31.77	3.00
10	2	0.00	0.00	25.30	3.14	29.12	0.00	31.44	2.95
11	1	45.21	45.51	0.00	0.00	0.00	26.75	0.00	0.00
12	6	45.30	45.51	0.00	0.00	0.00	26.72	0.00	0.00
V_{max}/K_M		13.78	13.85	13.03	13.11	12.59	14.99	11.43	13.83

EFFECT OF DUPLICATING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	13.21	13.27	13.69	14.19	12.07	14.37	10.96	14.91	82.59117
2	13.21	13.27	13.71	14.16	12.07	14.37	10.96	14.91	82.59287
3	13.21	13.27	13.71	14.17	12.07	14.37	10.96	14.91	82.59279
4	13.21	13.27	13.65	14.16	12.07	14.37	10.96	14.91	82.54874
5	13.21	13.27	13.64	14.23	12.07	14.37	10.96	14.86	82.55685
6	13.21	13.27	12.49	12.73	14.16	14.37	12.28	13.52	82.60657
7	13.21	13.27	12.49	12.71	13.34	16.01	12.04	13.48	82.72729
8	13.21	13.27	13.55	12.70	13.28	14.37	12.31	13.38	82.71636
9	13.21	13.27	13.55	12.70	13.29	14.37	12.28	13.38	82.69101
10	13.21	13.27	13.55	12.70	13.28	14.37	12.27	13.37	82.66692
11	15.09	15.17	12.49	12.57	12.07	15.48	10.96	13.25	82.50636
12	15.10	15.17	12.49	12.57	12.07	15.48	10.96	13.25	82.50766

EFFECT OF DELETING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step)								Growth Rate
	1	2	3	4	5	6	7	8	
1	14.41	14.48	12.32	11.94	13.16	15.67	11.95	12.65	82.52206
2	14.41	14.48	12.30	11.96	13.16	15.67	11.95	12.64	82.52225
3	14.41	14.48	12.30	11.96	13.16	15.67	11.95	12.64	82.52228
4	14.41	14.48	12.36	11.96	13.16	15.67	11.95	12.65	82.58177
5	14.41	14.48	12.37	11.89	13.16	15.67	11.95	12.70	82.56632
6	14.41	14.48	13.63	13.53	10.88	15.67	10.52	14.16	82.34006
7	14.41	14.48	13.63	13.55	11.77	13.88	10.77	14.20	82.52141
8	14.41	14.48	12.48	13.56	11.84	15.67	10.48	14.31	82.51709
9	14.41	14.48	12.48	13.56	11.83	15.67	10.51	14.32	82.55293
10	14.41	14.48	12.48	13.57	11.84	15.67	10.52	14.32	82.58632
11	12.35	12.41	13.63	13.71	13.16	14.46	11.95	14.45	82.53228
12	12.35	12.41	13.63	13.71	13.16	14.46	11.95	14.45	82.52982

Table 6.2.1

SAMPLED CELL AFTER 5000000 CELL DIVISIONS

Genome Size = 23
Genome Complexity = 19
Growth Rate = 83.19659

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	0.00	28.79	38.93	0.00	0.00	0.00	39.77
2	1	0.00	0.00	29.45	38.40	0.00	0.00	0.00	39.89
3	1	0.00	0.00	29.28	38.37	0.00	0.00	0.00	39.87
4	1	0.00	0.00	29.27	38.40	0.00	0.00	0.00	39.86
5	1	0.00	0.00	28.64	38.39	0.00	0.00	0.00	39.82
6	1	0.00	0.00	27.88	38.38	0.00	0.00	0.00	39.80
7	1	0.00	0.00	29.25	39.89	0.00	0.00	0.00	38.87
8	1	0.00	0.00	0.00	4.01	50.33	0.00	31.65	6.44
9	1	0.00	0.00	0.00	3.54	31.95	38.41	26.34	5.58
10	1	0.00	0.00	0.00	3.52	31.26	39.75	26.19	5.55
11	1	0.00	0.00	0.00	3.52	31.22	39.77	26.18	5.55
12	1	0.00	0.00	0.00	3.51	30.62	39.43	26.05	5.52
13	1	0.00	0.00	25.31	3.26	28.10	0.00	33.05	3.18
14	1	0.00	0.00	25.31	3.22	29.15	0.00	32.50	3.10
15	1	0.00	0.00	25.31	3.20	29.30	0.00	32.35	3.08
16	1	0.00	0.00	25.30	3.16	29.39	0.00	31.77	3.00
17	1	45.21	45.51	0.00	0.00	0.00	26.75	0.00	0.00
18	5	45.30	45.51	0.00	0.00	0.00	26.72	0.00	0.00
19	1	45.55	45.51	0.00	0.00	0.00	26.46	0.00	0.00
V_{max}/K_M		13.79	13.85	13.21	13.12	12.67	14.96	11.57	13.86

EFFECT OF DUPLICATING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	13.22	13.27	13.86	14.19	12.14	14.34	11.09	14.94	82.94068
2	13.22	13.27	13.88	14.17	12.14	14.34	11.09	14.95	82.94854
3	13.22	13.27	13.88	14.17	12.14	14.34	11.09	14.95	82.94203
4	13.22	13.27	13.88	14.17	12.14	14.34	11.09	14.95	82.94196
5	13.22	13.27	13.85	14.17	12.14	14.34	11.09	14.95	82.92209
6	13.22	13.27	13.82	14.17	12.14	14.34	11.09	14.95	82.89856
7	13.22	13.27	13.88	14.23	12.14	14.34	11.09	14.91	82.95859
8	13.22	13.27	12.66	12.74	14.24	14.34	12.41	13.56	82.93811
9	13.22	13.27	12.66	12.72	13.47	15.94	12.18	13.52	83.10751
10	13.22	13.27	12.66	12.72	13.44	16.00	12.18	13.52	83.10846
11	13.22	13.27	12.66	12.72	13.44	16.00	12.18	13.52	83.10711
12	13.22	13.27	12.66	12.72	13.41	15.98	12.17	13.52	83.07409
13	13.22	13.27	13.71	12.71	13.31	14.34	12.46	13.42	83.02442
14	13.22	13.27	13.71	12.70	13.35	14.34	12.44	13.42	83.03394
15	13.22	13.27	13.71	12.70	13.36	14.34	12.43	13.42	83.03208
16	13.22	13.27	13.71	12.70	13.36	14.34	12.41	13.41	83.00892
17	15.10	15.17	12.66	12.57	12.14	15.45	11.09	13.29	82.88665
18	15.11	15.17	12.66	12.57	12.14	15.45	11.09	13.29	82.88795
19	15.12	15.17	12.66	12.57	12.14	15.44	11.09	13.29	82.88809

Table 6.2.1

EFFECT OF DELETING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step)								Growth Rate
	1	2	3	4	5	6	7	8	
1	14.42	14.48	12.50	11.94	13.24	15.64	12.09	12.69	82.90902
2	14.42	14.48	12.47	11.97	13.24	15.64	12.09	12.68	82.90076
3	14.42	14.48	12.48	11.97	13.24	15.64	12.09	12.68	82.90980
4	14.42	14.48	12.48	11.97	13.24	15.64	12.09	12.68	82.90981
5	14.42	14.48	12.51	11.97	13.24	15.64	12.09	12.68	82.93669
6	14.42	14.48	12.54	11.97	13.24	15.64	12.09	12.69	82.96821
7	14.42	14.48	12.48	11.90	13.24	15.64	12.09	12.73	82.88308
8	14.42	14.48	13.81	13.53	10.95	15.64	10.66	14.20	82.75496
9	14.42	14.48	13.81	13.55	11.79	13.90	10.90	14.24	82.86815
10	14.42	14.48	13.81	13.55	11.82	13.84	10.90	14.24	82.86493
11	14.42	14.48	13.81	13.55	11.82	13.84	10.90	14.24	82.86674
12	14.42	14.48	13.81	13.55	11.85	13.85	10.91	14.24	82.91244
13	14.42	14.48	12.66	13.57	11.96	15.64	10.59	14.35	82.95732
14	14.42	14.48	12.66	13.57	11.92	15.64	10.62	14.35	82.94570
15	14.42	14.48	12.66	13.57	11.91	15.64	10.62	14.36	82.94849
16	14.42	14.48	12.66	13.57	11.91	15.64	10.65	14.36	82.98094
17	12.37	12.41	13.81	13.71	13.24	14.43	12.09	14.50	82.88051
18	12.36	12.41	13.81	13.71	13.24	14.43	12.09	14.50	82.87805
19	12.35	12.41	13.81	13.71	13.24	14.44	12.09	14.50	82.87546

Table 6.2.1

divisions, shown in figure 6.2.2f and (it *is* the same cell), the second part of table 6.2.1 (call these 'Cell 1' and 'Cell 2' respectively). It is quite clear that the proto-enzymes of each of these cells fall into five classes:

Table 6.2.2

Classes of Enzymes in Experiment 1

(a). Allocation of Protein

Class	*Activities*	*Cell 1*		*Cell 2*	
		Gene(s)	*Copies*	*Gene(s)*	*Copies*
I	3 4 8	1-5	7	1-7	7
II	4 5 7 8	6	1	8	1
III	4 5 6 7 8	7	4	9-12	4
IV	3 4 5 7 8	8-10	4	13-16	4
V	1 2 6	11-12	7	17-19	7

(b). Distribution of Activities

Class	*Cell 1*								*Cell 2*							
	% Total Activity (per Step)								*% Total Activity (per Step)*							
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
I	0	0	66	90	0	0	0	87	0	0	67	90	0	0	0	87
II	0	0	0	1	17	0	12	2	0	0	0	1	17	0	12	2
III	0	0	0	5	42	46	40	7	0	0	0	5	43	46	39	7
IV	0	0	34	4	41	0	48	4	0	0	33	4	40	0	49	4
V	100	100	0	0	0	54	0	0	100	100	0	0	0	54	0	0

The classes have arbitrarily been labelled I-V. It is obvious from table 6.2.2 that they and, indeed, many of the genes, are the same in the two cells. The relative representation of each class, shown under the columns headed 'Copies' in table 6.2.2a, is identical. All the cells explicitly reported during the latter half of the experiment showed this same allocation of protein between the classes described. The difference in growth rate of the two cells referred

to above, admittedly not large, is due to the introduction of new mutations, with resultant fine-tuning of the system. Only one of the twelve sequences (gene 5) in the earlier cell is not also represented among the nineteen of the later one.

As table 6.2.2b shows, the distribution of activities is almost identical. Class I carries almost all of the activity for the fourth and eighth metabolic steps, and two thirds of that for step three. Class V carries all of the activity for the first and second metabolic steps, and half the activity for the sixth. Classes III and IV share most of the load for steps five and seven, and respectively possess the remaining activity for steps six and three. The single gene representing class II in each cell (it is the same gene) is interesting. Its activity for step five is the highest activity in either cell though, being only single-copy, its contribution to that activity, and to step seven, is only modest. Of all possible deletions, however, deleting this gene is the most unfavourable in both cells. The two steps (five and seven) have the lowest net activities.

The absence of any favourable changes in genome size, and the apparent constraint on the system that mutations do not move the proto-enzyme 'out of its class' strongly suggests that a local, if not global, optimum has been reached. During the last one million cell divisions of the experiment, the mean growth rate fluctuated between 83.131 and 83.137, with the fastest cells in the population having growth rates a little under 83.20, as in the later cell described in the tables and in figure 6.2.1f.

Experiment 1

Figure 6.2.2

a) Extractable enzyme activities from individual cells sampled when stated.

Profile of the enzyme activities of the fastest cell sampled after:

b) 100,000 cell divisions.

c) 500,000 cell divisions.

d) 1,000,000 cell divisions.

e) 2,000,000 cell divisions.

f) 5,000,000 cell divisions.

(a).

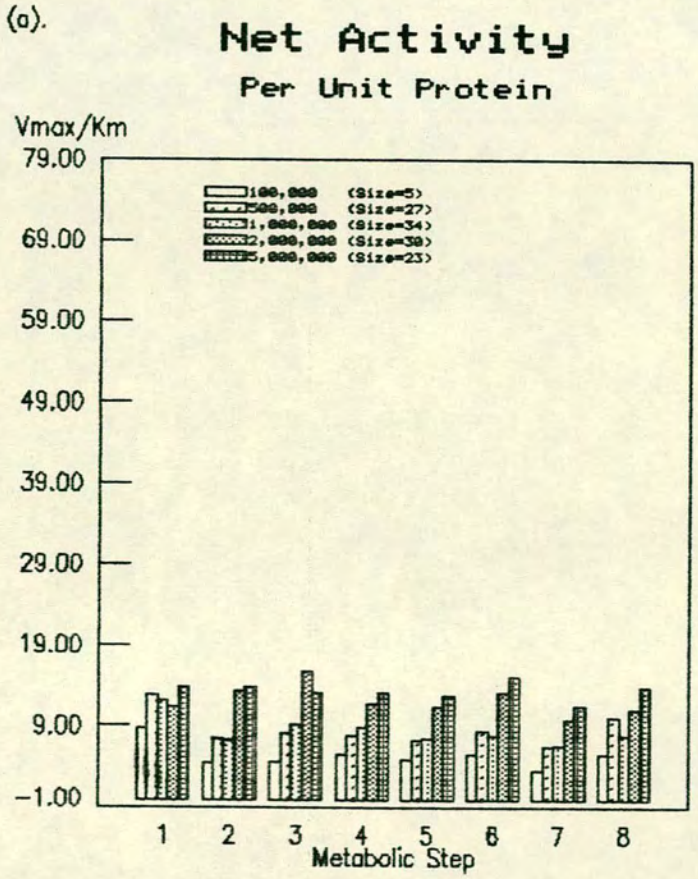
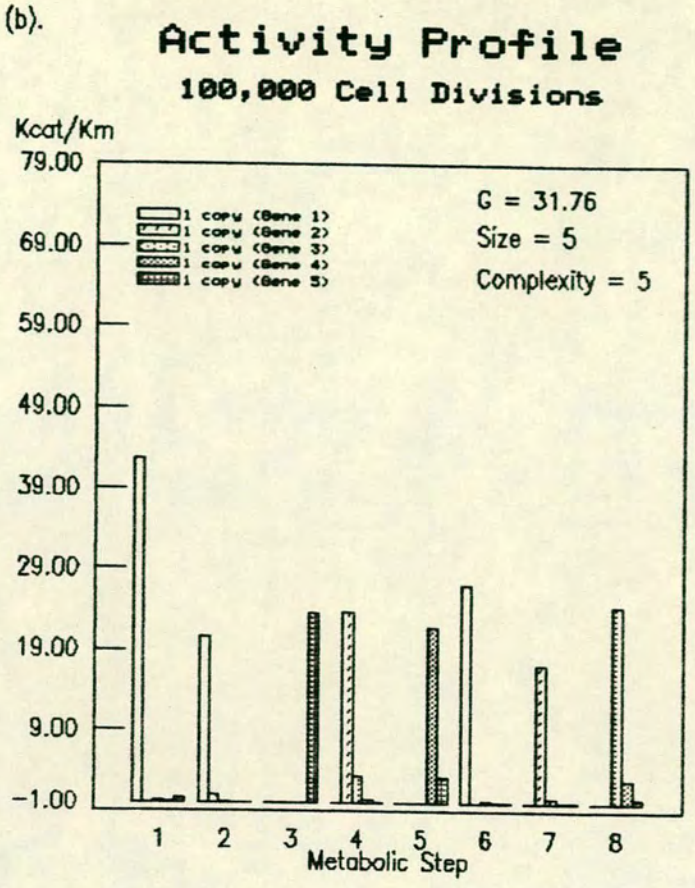
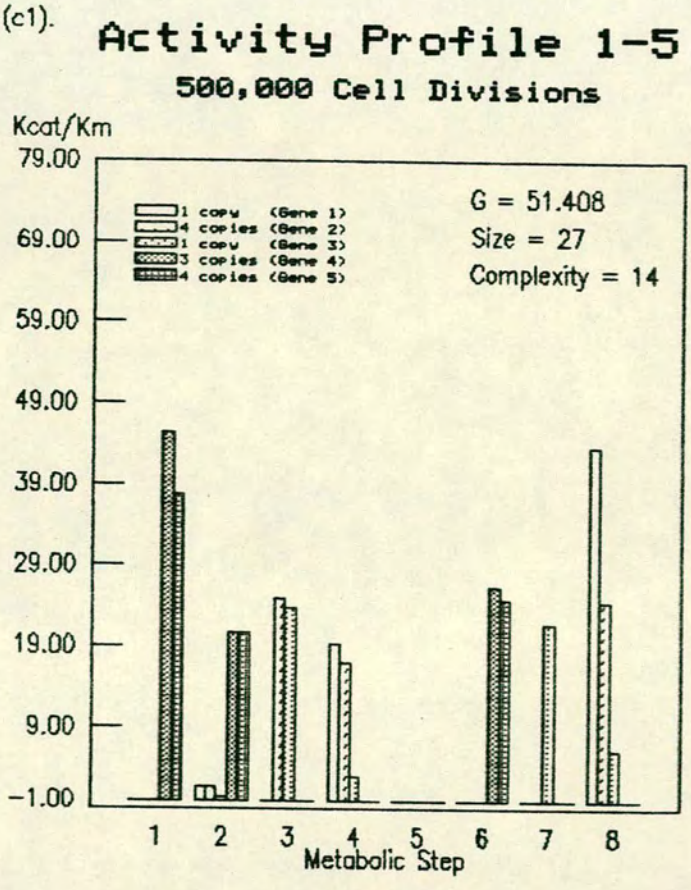


Figure 6.2.2

(b).

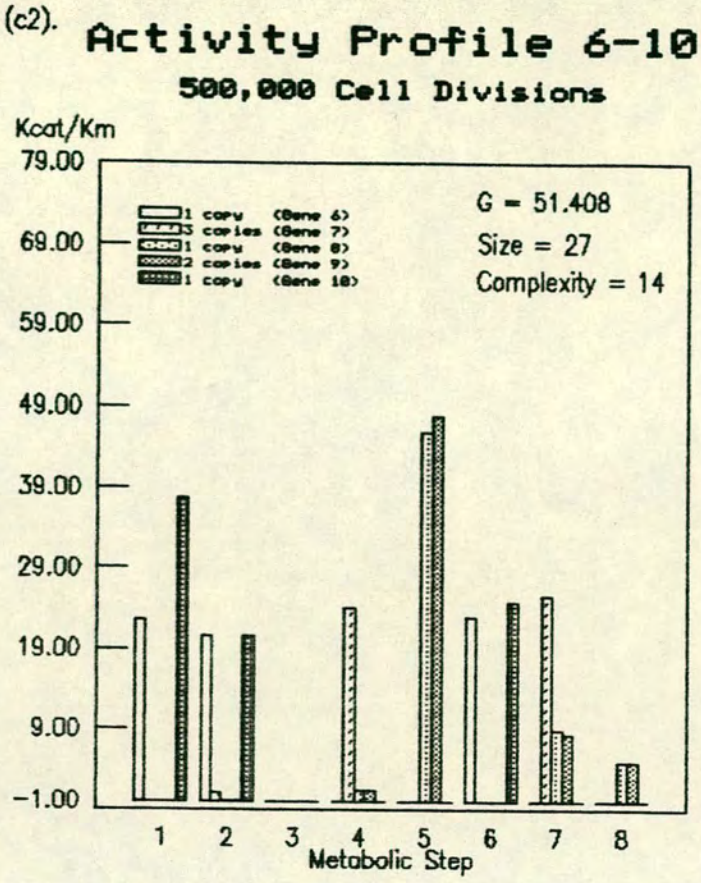


(c1).



igure 6.2.2

(c2).



(c3).

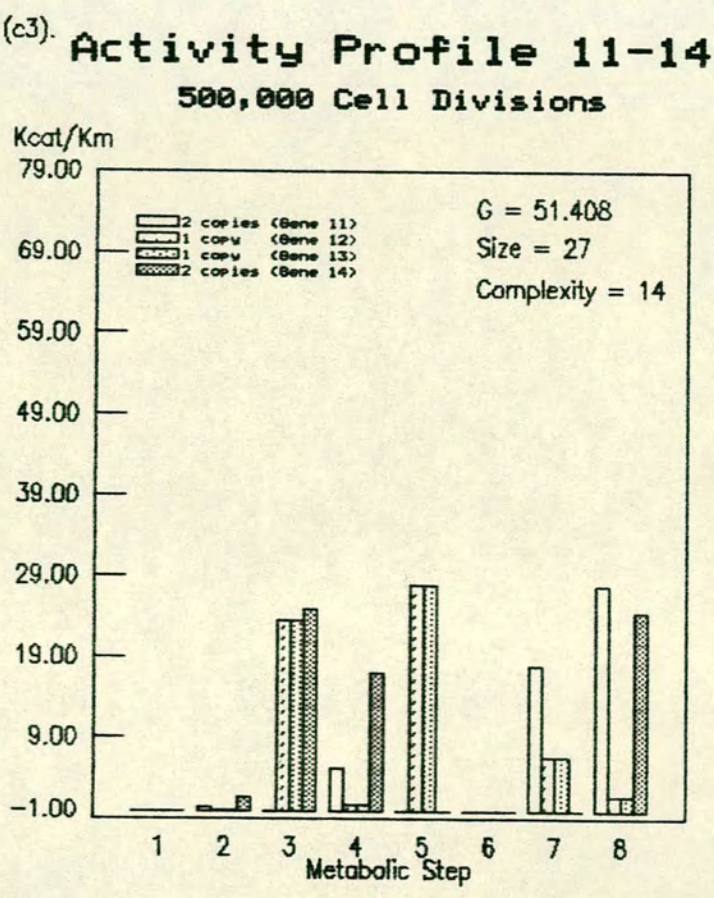
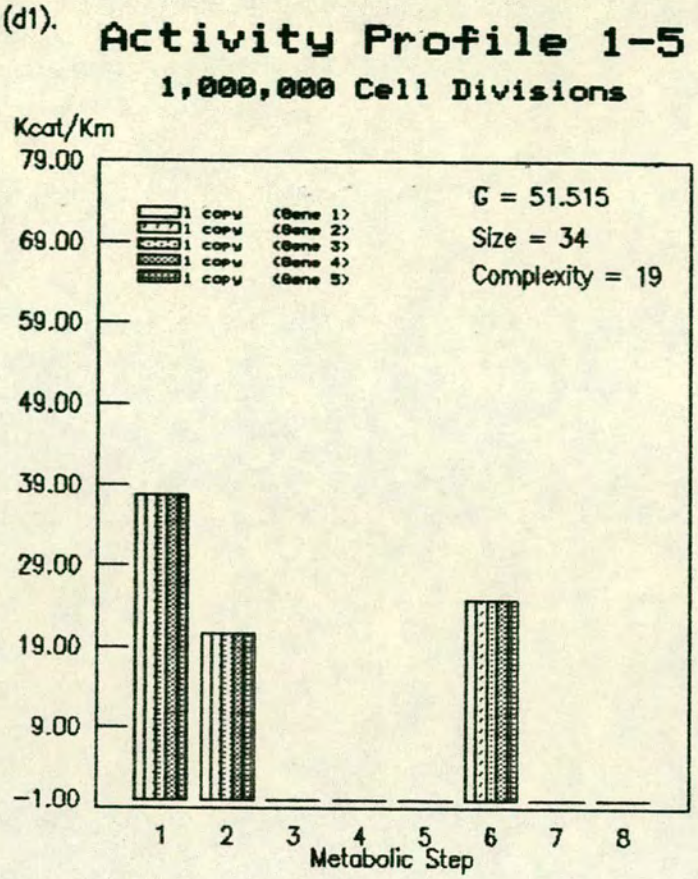


Figure 6.2.2

(d1).



(d2).

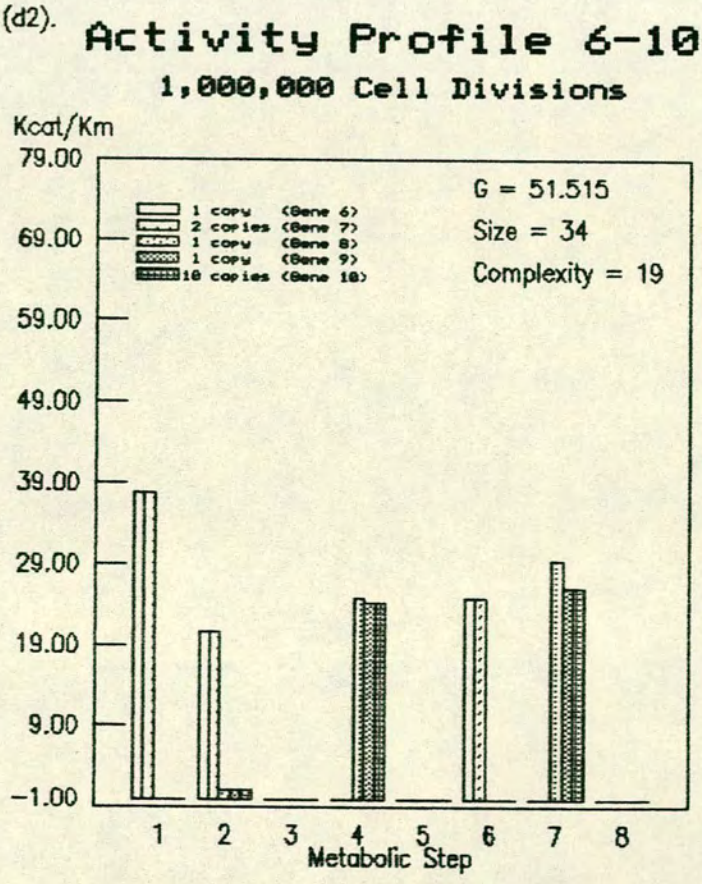
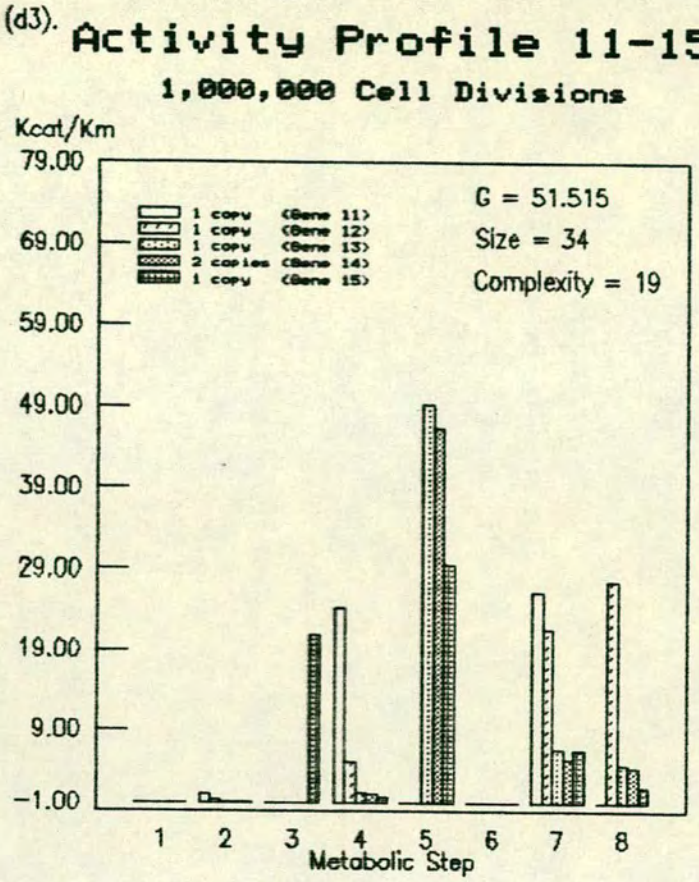


Figure 6.2.2

(d3).



(d4).

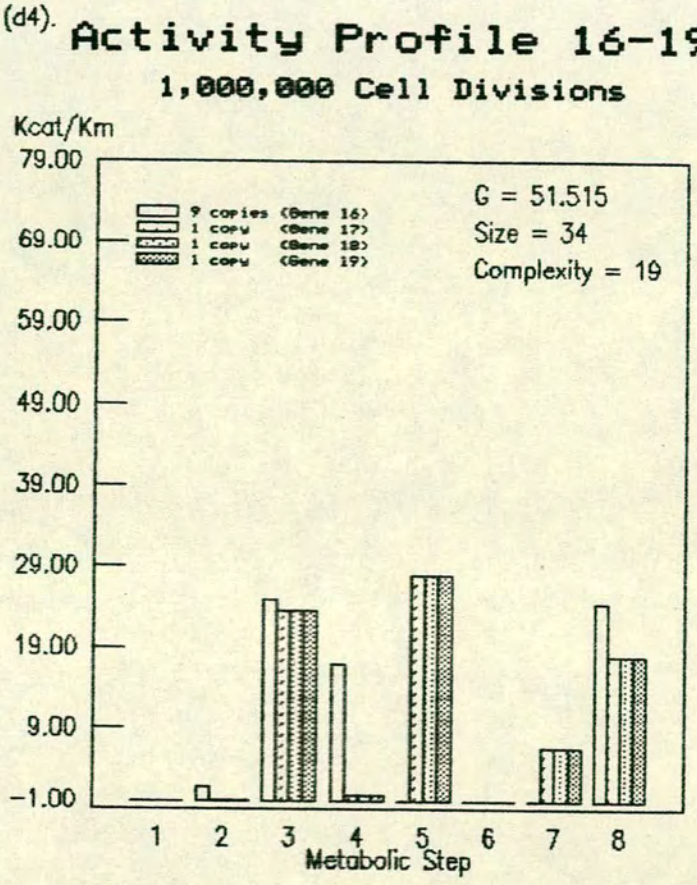
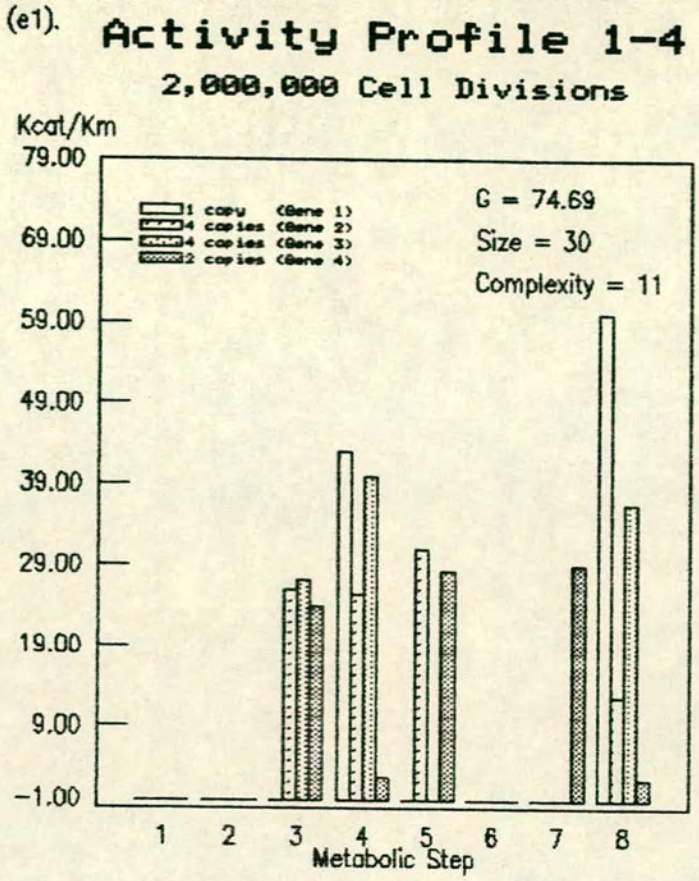


Figure 6.2.2

(e1).



(e2).

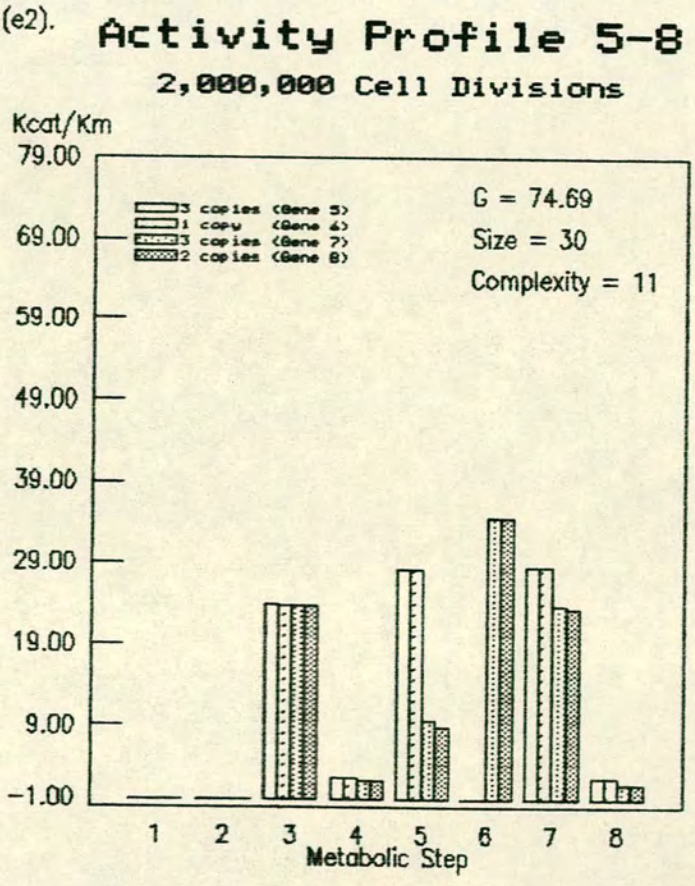


Figure 6.2.2

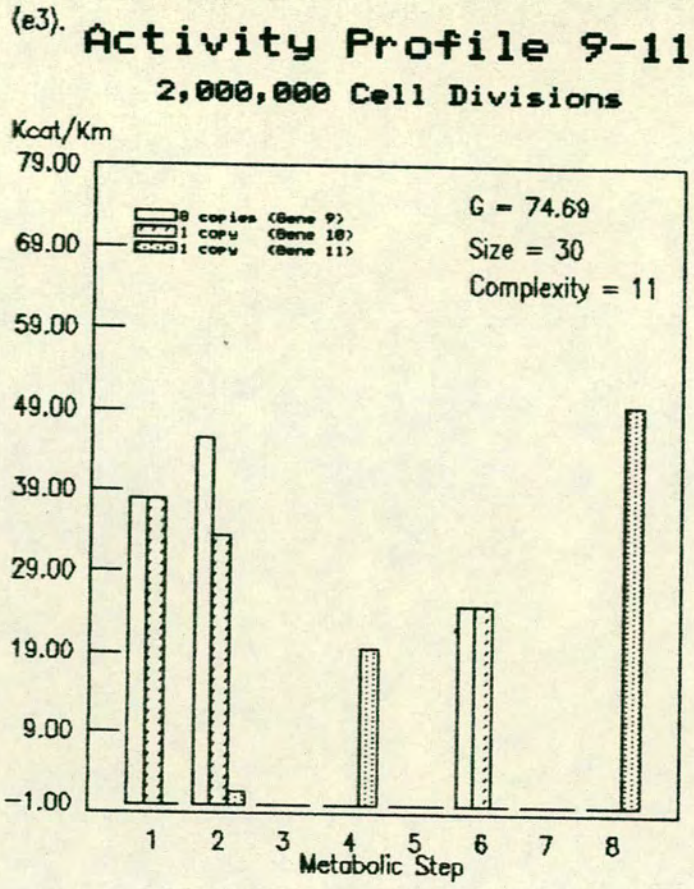
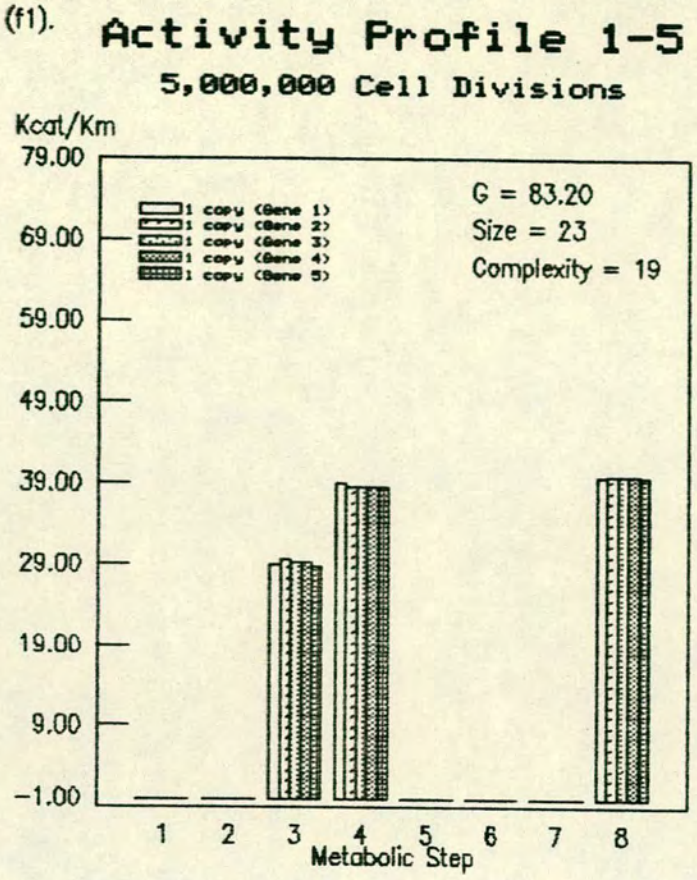

(e3).
Activity Profile 9-11
2,000,000 Cell Divisions
Kcat/Km
79.00
69.00
59.00
49.00
39.00
29.00
19.00
9.00
-1.00
0 copies (Gene 9)
1 copy (Gene 10)
1 copy (Gene 11)
G = 74.69
Size = 30
Complexity = 11
1
2
3
4
5
6
7
8
Metabolic Step

Figure 6.2.2

(f1).



(f2).

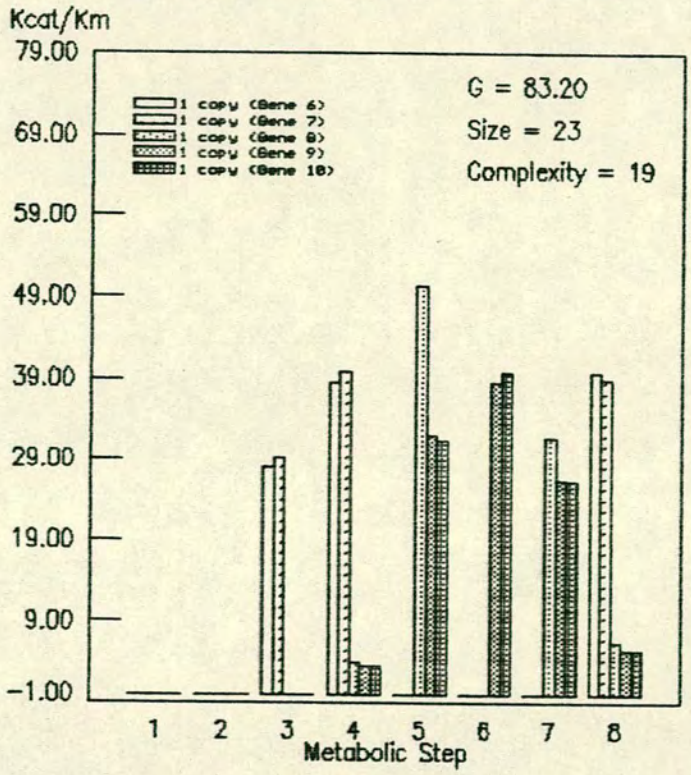
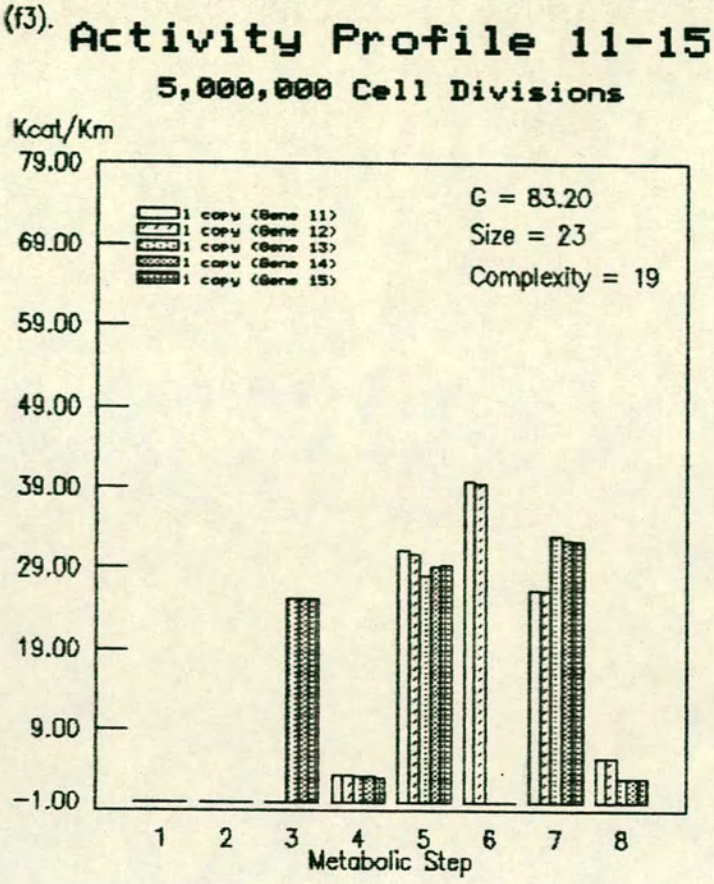


Figure 6.2.2

(f3).



(f4).

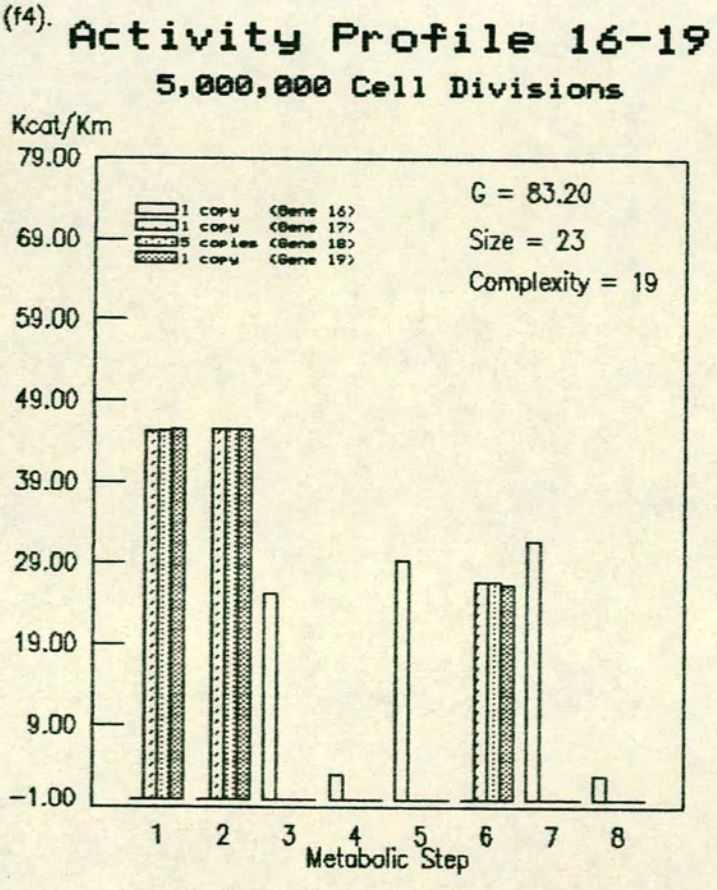


Figure 6.2.2

The average number of activities per enzyme (figure 6.2.1d) settled to around four, just as in the introductory experiment. The previous discussion gives a detailed breakdown of this.

One further lesson from the experiment, already alluded to, concerns the effect of the truncation factor on progress. The levelling off of the growth rate curve (figure 6.2.1b) at a growth rate of approximately 51 coincided with a period where the numbers of cells being rejected because of their low growth rates was falling (figure 6.2.1g). By far the majority of deaths were due to arbitrary kills because of overpopulation. The natural advantage of faster cells, which reproduce slightly more frequently than their sisters, was insufficient to guarantee their competitive triumph when fifty per cent of all cells were being killed without reference to performance, and when their numbers were initially small (favourable mutations presumably being a small subset of all mutations). Reducing the truncation factor produced a initial sharp rise in selective deaths, with accompanying reduction in deaths due to overpopulation. The number of cells selected against continued to increase as the mean growth rate climbed. As truncation selection began to take its toll, the number of arbitrarily killed cells fell, and the competitive edge of the faster cells became realizable. Eventually, the situation settled to one where the number of cells rejected by truncation selection, and killed randomly, was approximately equal. At that point the growth rate had settled to a relatively constant value.

The final mean growth rate in this experiment, 83.137, could clearly have been improved upon, as the fastest cells in the population had rates very close to

83.2. A further reduction in the truncation factor might, therefore, have been interesting. Two considerations suggest, however, that the maximum growth rate for this lineage is close to 83.2. Firstly, though hardly conclusively, the fastest cells over the last four hundred thousand cell divisions show no significant improvement. Secondly, it seems as though the 'solution' that the population has adopted can improve only by small 'fine-tuning' adjustments which are unlikely to make substantial impact on the performance.

Is the current solution a local peak or a global one? Can cells with growth rates substantially higher than 83.2 (for this particular pathway and pair of Lennard-Jones constants) be achieved? The mean activity for each step catalysed by Cell 2 is 13.38. If this were the activity for each of the eight steps, the growth rate of the cell would be 83.62, a fairly substantial increase in rate when set against the increase in growth rate actually achieved by the population over the last two million cell divisions of the experiment. The present solution has been to find an optimum allocation of protein between five classes of proto-enzyme. Unfortunately, a cell adopting this solution is trapped by it. The multifunctionality, and overlap, of the activities of its enzymes constrain it, so that improvements in the poorest activities cannot be achieved without incurring unacceptable loss of activity elsewhere. The multifunctional trap of Kacser & Beeby (1984) was described in terms of preventing the evolution of monofunctional proteins. In the present scenario, monofunctional proteins are not a good investment, but the cell is locked into a growth rate by a trap of just the same nature.

To describe the present cell as 'trapped' is, of course, to use an emotive word for a universal feature of all complex systems, natural and manmade. The cell is constrained by its history. The 'design decisions' made to obtain a fast cell are just that, 'decisions' to adopt one strategy rather than another. One uses the word, 'trapped', when there is evidence that leads one to believe that some of the decisions were the wrong ones: that another design, which is now unattainable, would have been better. In the present case, this evidence is not compelling, but an imagined cell with the same average activity as that under discussion is demonstrably faster. No evidence has been given to suggest that this cell can be physically realized given the particular substrates and physical laws that the system must use.

Early in the experiment there was a steep rise in genome size and complexity (figure 6.2.1c, and see also the abandoned experiment shown in figure 6.2.1h). Higher numbers of genes, as we saw in chapter 2, allow closer approximations to optimum protein allocation for some set of proto-enzymes than smaller numbers. The present simulations are generating the proteins as they progress, and the proto-enzyme composition is not initially static (we shall return to this later). In the present experiment, however, the composition is at least qualitatively fixed: there are five classes of proto-enzyme. After the next experiment, we shall address the question of whether there is a 'perfect' Class I (II, III ...) proto-enzyme, but before doing this, a simple experiment is to ask how closely the present gene composition of, say, Cell 2, approximates the optimum. Of course, that optimum cannot be calculated using the simple methods of chapter 2, because the proto-enzymes overlap in activity, and so the allocation of protein to each activity is not independent of all the

others. However, we can take the present cell and, maintaining its composition, increase its genome size and ask if there are then any favourable duplications or deletions now present. This will not generally give us the optimum, but may indicate whether significant improvement on the archetypal twenty-three gene, five-class cell can be made.

The later sampled cell depicted in table 6.2.1 and decribed as 'Cell 2' in table 6.2.2 was taken as the subject of this exercise. Each of its genes (and hence its genome size) was increased in number by a factor of five. This, of course, has no effect on its growth rate (table 6.2.3). However, when attempts are made to introduce duplications or deletions into the cell, its response is very different from cell 2, in which all such changes result in slower growth.

Table 6.2.3 shows that the genome-expanded cell has an increased growth rate if one of the copies of gene 6 (coding for a class I proto-enzyme) is deleted, or if the number of copies of any gene coding for a class III proto-enzyme (genes 9-12) is increased by duplication. The process is repeated by taking the fastest growing of these potential descendants (in this case, the cell in which gene 10 is duplicated) and asking what favourable single gene copy changes there are. In this way, the exercise was continued until a cell was obtained which, like cell 2, has no favourable (single) duplications or deletions. It is rather surprising that more than sixty iterations of this procedure were gone through before the final cell was reached. The steps, and the final cell, are given in table 6.2.3. The total genome size increases by only one, but there is a considerable redistribution of protein, with proto-enzymes being eliminated completely.

Table 6.2.3

CELL 2 WITH ALL GENE DOSAGES INCREASED FIVE-FOLD

Genome Size = 115
Genome Complexity = 19
Growth Rate = 83.19659

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	0.00	0.00	28.79	38.93	0.00	0.00	0.00	39.77
2	5	0.00	0.00	29.45	38.40	0.00	0.00	0.00	39.89
3	5	0.00	0.00	29.28	38.37	0.00	0.00	0.00	39.87
4	5	0.00	0.00	29.27	38.40	0.00	0.00	0.00	39.86
5	5	0.00	0.00	28.64	38.39	0.00	0.00	0.00	39.82
6	5	0.00	0.00	27.88	38.38	0.00	0.00	0.00	39.80
7	5	0.00	0.00	29.25	39.89	0.00	0.00	0.00	38.87
8	5	0.00	0.00	0.00	4.01	50.33	0.00	31.65	6.44
9	5	0.00	0.00	0.00	3.54	31.95	38.41	26.34	5.58
10	5	0.00	0.00	0.00	3.52	31.26	39.75	26.19	5.55
11	5	0.00	0.00	0.00	3.52	31.22	39.77	26.18	5.55
12	5	0.00	0.00	0.00	3.51	30.62	39.43	26.05	5.52
13	5	0.00	0.00	25.31	3.26	28.10	0.00	33.05	3.18
14	5	0.00	0.00	25.31	3.22	29.15	0.00	32.50	3.10
15	5	0.00	0.00	25.31	3.20	29.30	0.00	32.35	3.08
16	5	0.00	0.00	25.30	3.16	29.39	0.00	31.77	3.00
17	5	45.21	45.51	0.00	0.00	0.00	26.75	0.00	0.00
18	25	45.30	45.51	0.00	0.00	0.00	26.72	0.00	0.00
19	5	45.55	45.51	0.00	0.00	0.00	26.46	0.00	0.00
V_{max}/K_M		13.79	13.85	13.21	13.12	12.67	14.96	11.57	13.86

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
116	19	Duplicate	10	83.20824	9 11 12;	6
115	19	Delete	16	83.21002		
116	19	Duplicate	14	83.21398	9 10 11 13 15;	17 18 19
115	19	Delete	16	83.21595		
116	19	Duplicate	14	83.21971	9 10 11 13 15;	17 18 19
115	19	Delete	16	83.22188		
116	19	Duplicate	14	83.22543	9 10 11 13 15;	17 18 19
115	19	Delete	16	83.22779		
116	19	Duplicate	14	83.23115	9 10 11 13 15;	17 18 19
115	18	Delete	16	83.23370		
116	18	Duplicate	10	83.23686	9 11 13 14 15;	17 18 19
117	18	Duplicate	7	83.24262	1 2 3 4;	12
116	18	Delete	6	83.25153		5
117	18	Duplicate	7	83.25677	1 2 3 4;	12
116	18	Delete	6	83.26615		5
117	18	Duplicate	7	83.27088	1 2 3 4;	12
116	18	Delete	6	83.28074		5
115	18	Delete	12	83.28519	2 3 4 7;	
116	18	Duplicate	10	83.28845	9 11 13 14 15;	17 18 19
115	18	Delete	12	83.29316	1 2 3 4 7;	
116	18	Duplicate	10	83.29617	9 11 13 14 15;	17 18 19
115	18	Delete	12	83.30112	1 2 3 4 7;	
116	18	Duplicate	10	83.30388	9 11 14 15;	17 18 19
115	18	Delete	12	83.30908	1 2 3 4 7;	
116	18	Duplicate	10	83.31159	9 11 14 15;	17 18 19
115	17	Delete	12	83.31704	1 2 3 4 7;	
116	17	Duplicate	10	83.31929	9 11 14 15;	17 18 19
117	17	Duplicate	7	83.32377	1 2 3 4;	
116	17	Delete	6	83.33385		5
117	17	Duplicate	7	83.33782	2 3 4;	
116	16	Delete	6	83.34837		1 3 4 5
117	16	Duplicate	7	83.35182	2;	
116	16	Delete	5	83.35721		1 3 4
117	16	Duplicate	7	83.36033	2;	
116	16	Delete	5	83.36603		1 3 4
117	16	Duplicate	7	83.36881	2;	
116	16	Delete	5	83.37481		1 3 4
117	16	Duplicate	7	83.37726	2;	
116	16	Delete	5	83.38357		1 3 4
117	16	Duplicate	7	83.38568		
116	15	Delete	5	83.39230		1 2 3 4
117	15	Duplicate	7	83.39408		
116	15	Delete	1	83.39652		2 3 4
117	15	Duplicate	7	83.39814		
116	15	Delete	1	83.40072		2 3 4

contd next page...

Table 6.2.3

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
117	15	Duplicate	7	83.40219		
116	15	Delete	1	83.40490		2 3 4
117	15	Duplicate	7	83.40623		
116	15	Delete	1	83.40907		2 3 4
117	15	Duplicate	7	83.41024		
116	14	Delete	1	83.41322		2 3 4
117	14	Duplicate	7	83.41425		
116	14	Delete	4	83.41706		2 3
117	14	Duplicate	7	83.41790		
116	14	Delete	4	83.42087		2 3
117	14	Duplicate	7	83.42153		
116	14	Delete	4	83.42465		2 3
117	14	Duplicate	7	83.42514		
116	14	Delete	4	83.42840		2 3
117	14	Duplicate	7	83.42871		
116	13	Delete	4	83.43213		2 3
117	13	Duplicate	7	83.43227		
116	13	Delete	3	83.43580		2
116	13	None	0	83.43580		

Table 6.2.3

FINAL CELL WITH NO FAVOURABLE SINGLE DUPLICATIONS/DELETIONS

Genome Size = 116
Genome Complexity = 13
Growth Rate = 83.43580

(Genes are numbered as in Cell 2 to facilitate comparison)

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	0								
2	5	0.00	0.00	29.45	38.40	0.00	0.00	0.00	39.89
3	4	0.00	0.00	29.28	38.37	0.00	0.00	0.00	39.87
4	0								
5	0								
6	0								
7	26	0.00	0.00	29.25	39.89	0.00	0.00	0.00	38.87
8	5	0.00	0.00	0.00	4.01	50.33	0.00	31.65	6.44
9	5	0.00	0.00	0.00	3.54	31.95	38.41	26.34	5.58
10	12	0.00	0.00	0.00	3.52	31.26	39.75	26.19	5.55
11	5	0.00	0.00	0.00	3.52	31.22	39.77	26.18	5.55
12	0								
13	5	0.00	0.00	25.31	3.26	28.10	0.00	33.05	3.18
14	9	0.00	0.00	25.31	3.22	29.15	0.00	32.50	3.10
15	5	0.00	0.00	25.31	3.20	29.30	0.00	32.35	3.08
16	0								
17	5	45.21	45.51	0.00	0.00	0.00	26.75	0.00	0.00
18	25	45.30	45.51	0.00	0.00	0.00	26.72	0.00	0.00
19	5	45.55	45.51	0.00	0.00	0.00	26.46	0.00	0.00
V_{max}/K_M		13.67	13.73	12.98	13.29	12.86	15.53	11.68	13.65

Table 6.2.3

How different is the final cell from the original one? All five proto-enzyme classes are still present. The relative allocation of protein between classes has changed a little, standing at 35:5:22:19:35 in the final cell compared with 35:5:20:20:35 (7:1:4:4:7) in the 'polyploid' cell 2. It seems, therefore, that the equal allocation of protein to classes III and IV in cells with twenty three genes is an approximation to a distribution slightly favouring class III.

The most significant change, as indicated above, is the allocation of protein within classes. Four of the seven original genes comprising class I have been deleted, with the allocation going into gene 7, which has the highest activity for the relatively improvished fourth metabolic step, to which class I proto-enzymes are the major contributors. Less dramatically, class III has lost gene 12, and invested heavily in gene 10 (increasing, also, its total share of protein). The loss of gene 12 is not surprising, as for each of the five steps catalysed by the class, it had the least activity. In a parallel way (but with a slight loss in net allocation), class IV has lost gene 16, diverting its resources into gene 14. Classes II and V are unchanged, though they have lost nearly 1 per cent of their previous allocation because of the increase in genome size.

The mean net activity of the final cell is 13.42, about 0.3 per cent higher than the 13.38 of cell 2, and this is also the approximate gain in growth rate (it should be apparent to the reader that increases in the mean activity can have negligible, indeed adverse, effect on growth rate: it depends critically on distribution).

The distribution of activities among proto-enzymes clearly depends on the similarity of the substrates. The ancestral enzyme, which has all eight activities, has higher activity towards the reactions with transition states with at least one relatively large dimension, because all three of its axes are large. As the population evolves, enzymes with substantially higher activities arise, which are obtained by tailoring the active site so that it more closely fits some substrates, while sterically excluding others.

The dimensions of the pathway were shown in figure 6.1.2a and are repeated in figure 6.2.3a. The second part of that figure (6.2.3b) shows the dimensions of one proto-enzyme from each class of cell 2 (class I: gene 7; II: 8; III: 10; IV: 14; V: 18). In fact, the different proto-enzymes of each class tend to differ in only one of their three dimensions, and usually for that dimension are the same to three to five significant digits, and so would be identical on the scale of the figure.

The distribution of activities is now fairly readily interpreted. The transition states for reactions 1, 2 and 6 share in common that they have a small y-dimension, and one other large dimension. The class V proto-enzymes catalyse these three steps by having a small y-dimension (excluding other reaction intermediates from the active site), and large x- and z-dimensions. This means that, for example, the binding energy between the x-dimension of the transition step for reaction 1, and the class V active site x-dimension is low, but the binding in the other two dimensions is high.

Enzymes of classes II and III differ in only one of their dimensions (*z*). The smaller site of class II explains the fact that while III has high activity for reaction 6, II has none. In 'compensation', II has higher activity than III for all of III's other substrates. The first three metabolic steps are not catalysed by these two classes because each of these steps has one very large axis (*z*, *x*, and *y* respectively). Classes I and IV, however, have large *y* axes that can accomodate the *y*-axis of the transition state for the third metabolic step. The class IV proto-enzymes have larger *x* and *z* axes than those of class I accounting for the difference between the classes in respect of their ability to catalyse steps 5 and 7. Again there is a trade-off between quality and quantity, with the activity of class I being higher than class III for all activities they share in common.

The fourth and eighth metabolic steps have no particularly large or small dimensions, and are catalysed by proto-enzymes of all classes except I (which, as we have seen, has a small active site *y*-axis).

Experiment 1;

Figure 6.2.3

a) The metabolic pathway of experiment 1, showing transition state dimensions, and the value of Rmax (distance between entities giving highest interaction energy).

This figure is the same as 6.1.2a but is repeated here to allow comparison with (b).

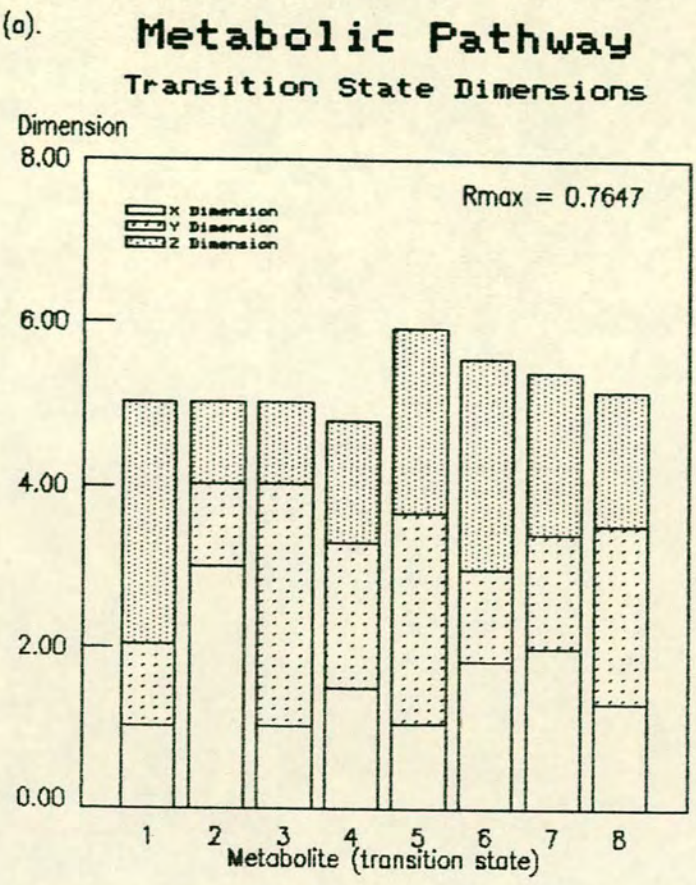
b) The "sequences" of the five classes (I–V: numbered 1–5 in the chart) of enzymes of the final population.

The y–axis is scaled so that an axis dimension of 2 in (a) has the same physical height as its maximally binding dimension (2.7647) in this figure.

For smaller dimensions the active site is 'taller', and for larger dimensions, smaller, than the corresponding dimension in (a).

Figure 6.2.3

(a).



(b).

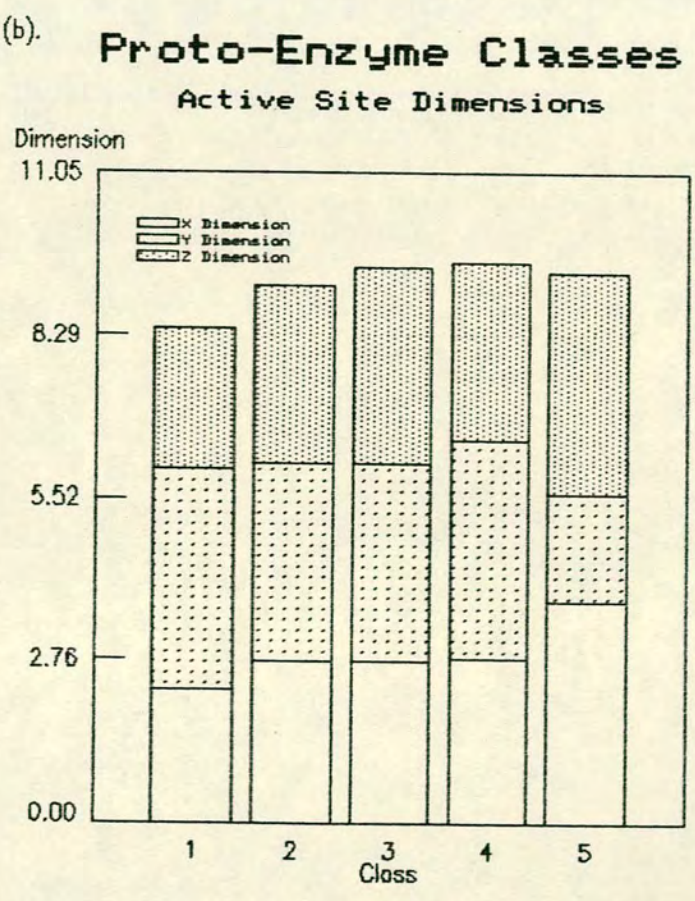


Figure 6.2.3

Pathway Favouring Multifunctionality - Experiment 2

In chapter three we examined the question of overlap in enzyme activities (One Reaction-Many Enzymes). The final population of experiment 1 was comprised of cells with gene products all of which had some overlap in their activities. Even when the products are grouped into classes there were only two reactions (the first and second of the pathway) which were catalysed by a single class of proto-enzyme, the remaining six reactions being supported by at least two (as many as four in some cases). The present pathway and Lennard-Jones constants do not favour monofunctionality, but this does not preclude the possibility that multifunctional gene products with minimal overlap in their activities may give high growth rates. The high cost of monofunctionality lies in the fact that the gain in activity per catalytic species is not sufficiently high to offset the requirement to produce eight different proto-enzyme species (classes if you like). These costs are not incurred by a solution with monocatalysed reactions and multifunctional proto-enzymes, and the present experiment examines a population which went some way towards adopting this strategy.

This experiment, for convenience call it Experiment 2, is a repeat of the introductory and first experiments. The pathway and Lennard-Jones constants are the same. The progress of the simulation is shown in figure 6.3.1. It comprises only two runs (figure 6.3.1a), because the second run was prematurely terminated by the extinction of the population.

The early extinction of the population was due to the effect of truncation selection (figure 6.3.1g). The truncation factor for the experiment was 1.0, a decision influenced by the results we have just discussed (recall figure 6.2.1b). After the first 350,000 cell divisions or so, during which there is one brief drop in population size, fatalities due to truncation selection were considerably less than those due to overpopulation, but over the last 20,000 cell divisions, there was a sharp rise in selective deaths. The reason for this is clear from figure 6.3.1e. Towards the end of the experiment, there is a sharp rise in the variation in growth rate in the population. Presumably a small number of cells with growth rates well above the mean arose. As a result, the mean growth rate exceeded that of almost all cells in the population. A truncation factor of 1.0 means that cells which have growth rates less than the mean of the previous sample are selected against. The small number of cells with higher than average rates must have been killed before division by 'random' deaths due to overpopulation (recall that only one of the daughters of a cell division are liable to mutational events in the current scenario).

The mean growth rate rises to a little over 76 (figure 6.3.1b) and remains at this level throughout the experiment. This is somewhat less than the rate at which progress levelled off in either of the previous experiments with this pathway.

Figure 6.3.1f reveals that all variation in genome size and complexity in the population disappeared after some 200,000 cell divisions. In fact, both the size and complexity of genomes, after an initial rise, settle at the surprising

Experiment 2;

Duration: 1.08 million cell divisions.
(Terminated by extinction).

Sampled: Every 3000 cell divisions.

Truncation factor: 1.00

Pathway and ancestral cell are
as in the introductory experiment.

Figure 6.3.1

a) Population: reduced to 100 at end of each session.

b) Mean growth rate of population.

c) Mean genome size & complexity.

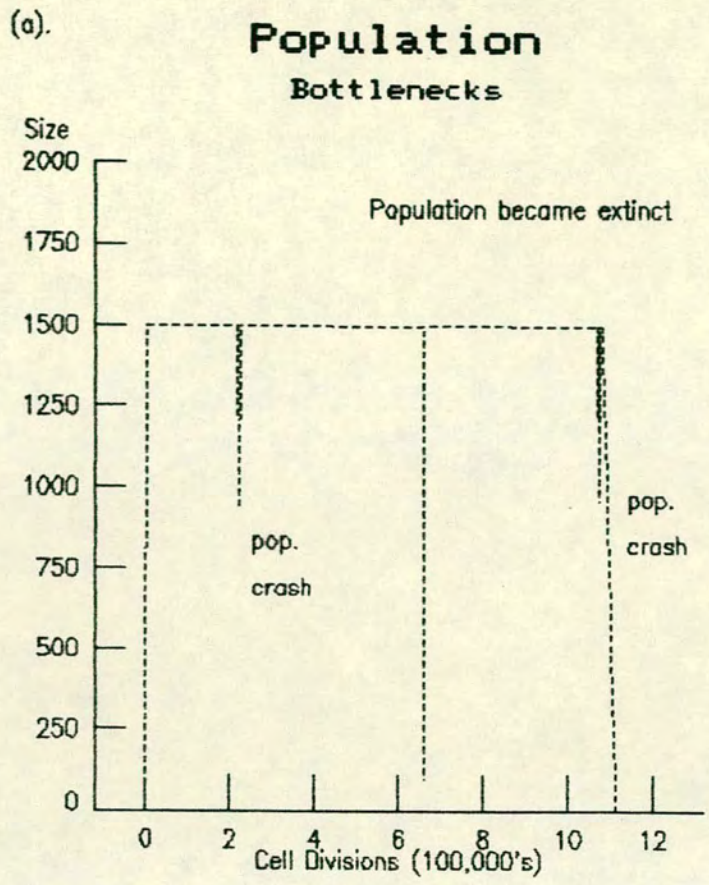
d) Mean number of reactions catalysed by each enzyme.

e) Variation in growth rate.

f) Variation in size & complexity.

g) Cell deaths from all causes.

Figure 6.3.1


(a).
Population
Bottlenecks
Size
2000
1750
1500
1250
1000
750
500
250
0
Population became extinct
pop.
crash
pop.
crash
0
2
4
6
8
10
12
Cell Divisions (100,000's)

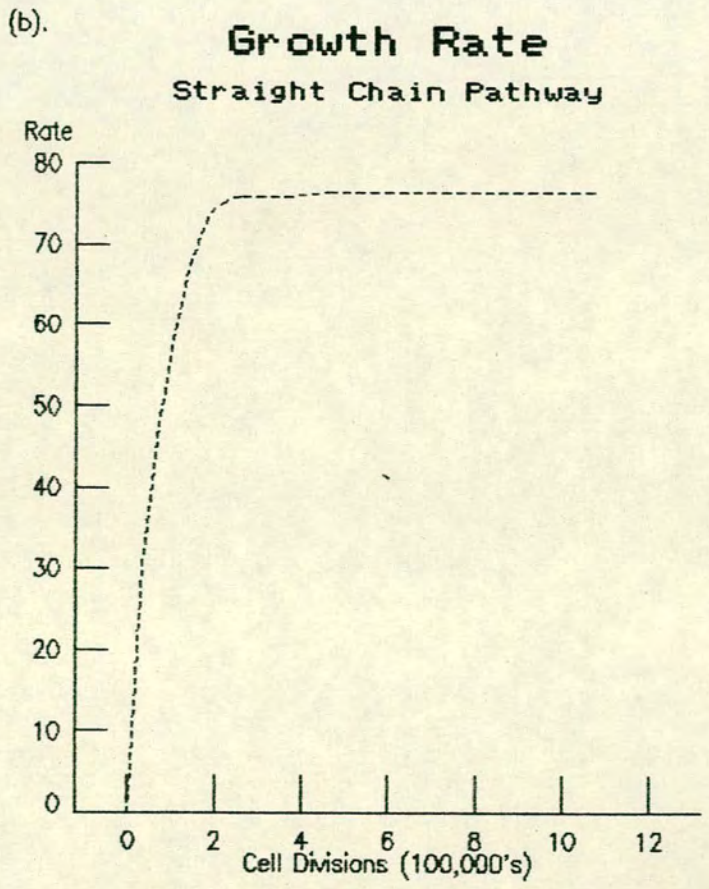
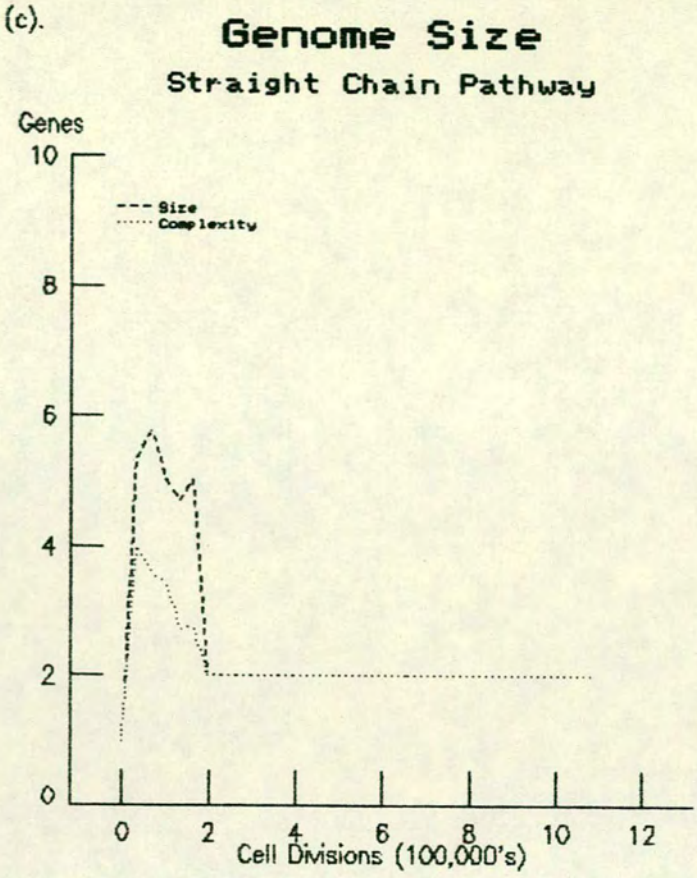

(b).
Growth Rate
Straight Chain Pathway
Rate
80
70
60
50
40
30
20
10
0
0
2
4
6
8
10
12
Cell Divisions (100,000's)

Figure 6.3.1

(c).



(d).

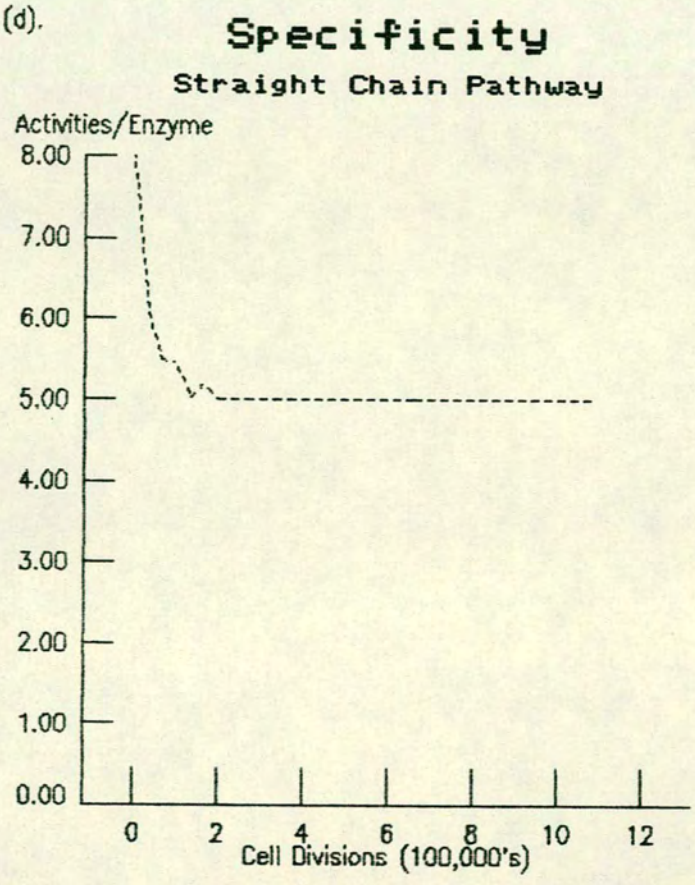
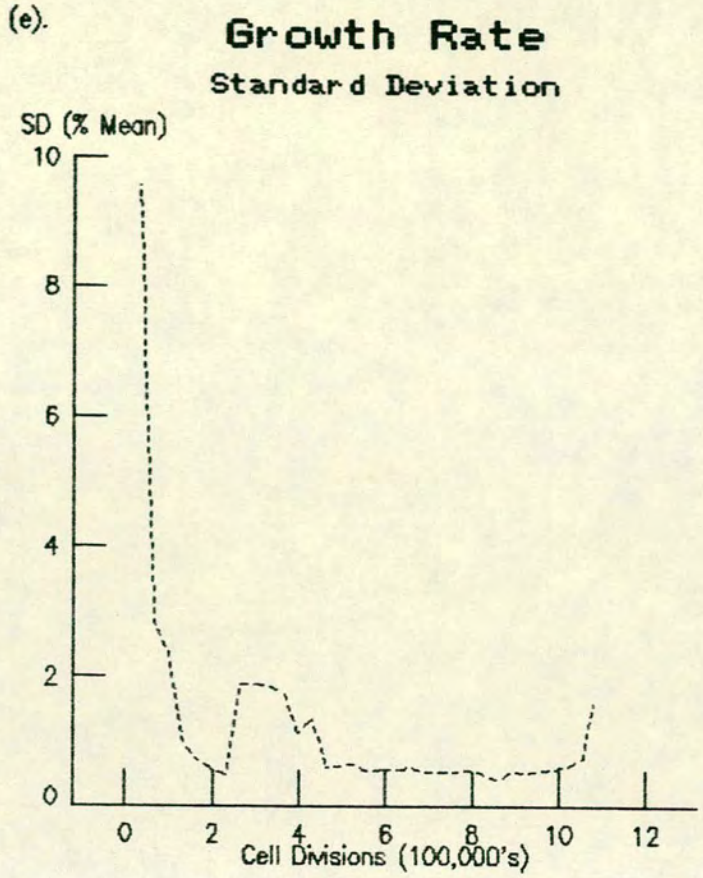


Figure 6.3.1

(e).



(f).

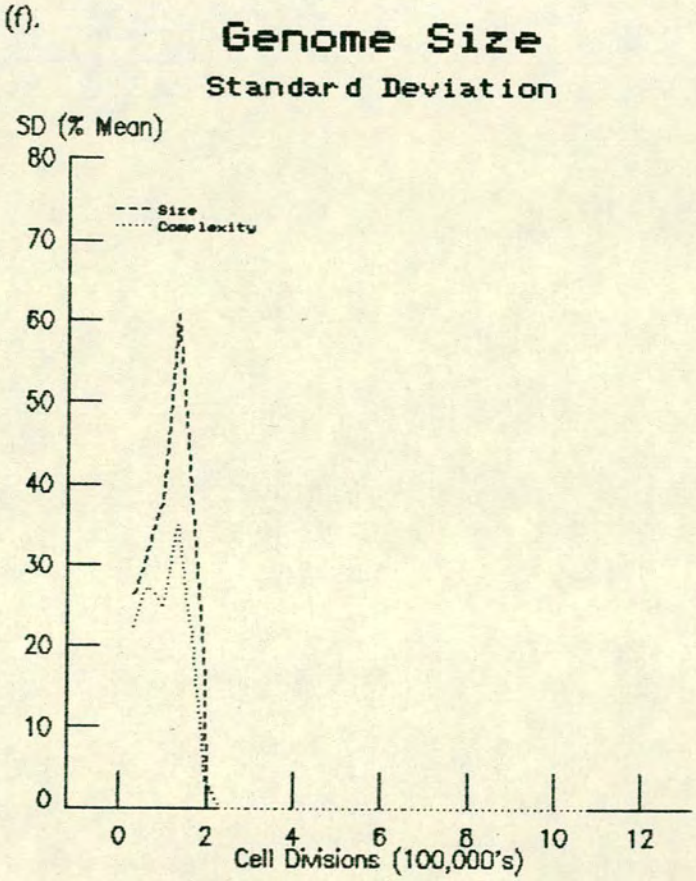


Figure 6.3.1

(g).

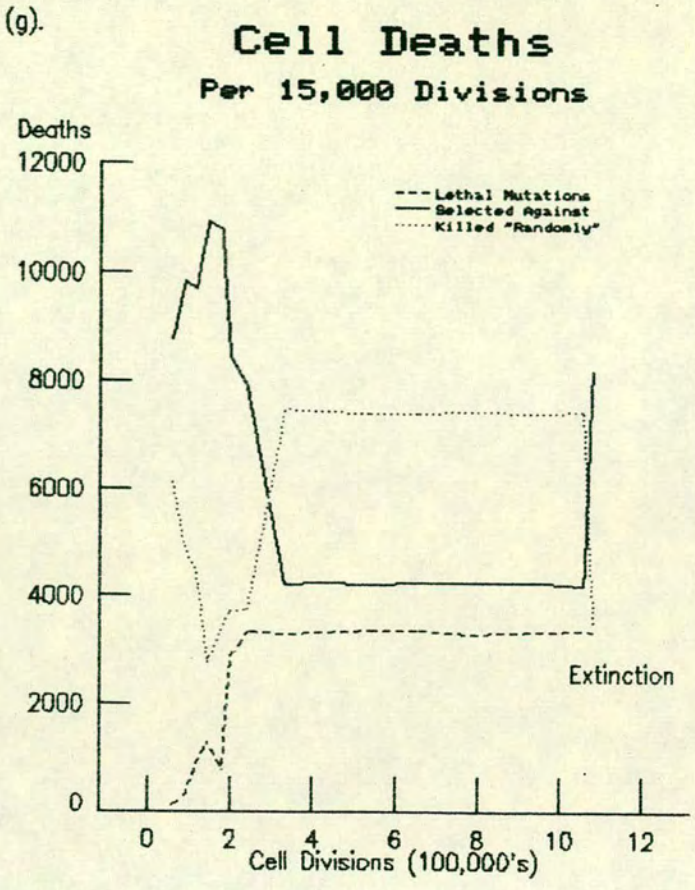

Cell Deaths
Per 15,000 Divisions
Deaths
12000
10000
8000
6000
4000
2000
0
Lethal Mutations
Selected Against
Killed "Randomly"
Extinction
0
2
4
6
8
10
12
Cell Divisions (100,000's)

Figure 6.3.1

value of two (figure 6.3.1c). It is clear, of course, that if the average number of activities per proto-enzyme is about four (recall figures 6.1.2b and 6.2.1d), and there are eight metabolic steps to catalyse, then two proto-enzymes are enough to support growth, even at the levels of activity evolved in the introductory and first experiments. However, it should be recalled that no enzyme we have yet considered has had high activity for more than three steps. In experiment 1, for example, a viable cell may be formed by taking a single class IV and a class V proto-enzyme, and table 6.3.1 shows such a cell. Its growth rate is low, which is not surprising in view of the low activity of the class IV proto-enzyme for the fourth and eighth steps. The situation is improved a little if the gene coding for this enzyme is duplicated a couple of times, but the growth rate remains below 40.0 even then, and further duplications of the gene reduce growth rate because of the reduction in the net activities supported by the class V gene product (in particular, the sixth step). Duplication cannot, in any event, remedy the inherently poor catalytic activity for steps four and eight, which is due to the structure, not the concentration, of the catalyst.

In the present experiment, the average number of activities per catalyst is not four, but five (figure 6.3.1d). Also unlike the previous experiments, this number does not vary with time: the mean remains at exactly five catalytic activities per proto-enzyme. This means that, unlike the cell of table 6.3.1, which has only a single enzyme catalysing each step, two of the steps must be catalysed by both enzymes (it cannot be more than two, or there would be an uncatalysed reaction and the cell would be dead).

Table 6.3.1

CELL WITH 1 CLASS IV AND 1 CLASS V GENE

Genome Size = 2
Genome Complexity = 2
Growth Rate = 30.51405

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	0.00	25.31	3.22	29.15	0.00	32.50	3.10
2	1	45.30	45.51	0.00	0.00	0.00	26.72	0.00	0.00
V_{max}/K_M		22.65	22.76	12.65	1.61	14.57	13.36	16.25	1.55

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
3	2	Duplicate	1	37.00530		
4	2	Duplicate	1	38.17770		
4	2	None	0	38.17770		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 4
Genome Complexity = 2
Growth Rate = 38.17770

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	3	0.00	0.00	25.31	3.22	29.15	0.00	32.50	3.10
2	1	45.30	45.51	0.00	0.00	0.00	26.72	0.00	0.00
V_{max}/K_M		11.32	11.38	18.98	2.41	21.86	6.68	24.38	2.33

The kinetic features of cells sampled at various points in the experiment are presented in figure 6.3.2. The net activity profile (figure 6.3.2a) shows that after 200,000 cell divisions (by which point all cells in the population have a genome size and complexity of two) the fourth and sixth metabolic steps have the lowest activities, with step seven, which was the lowest net activity at the end of experiment 1, doing slightly better and showing increasing activity at each sampled point.

The fastest cell sampled after 66,000 cell divisions had a genome size and complexity of three (figure 6.3.2b). The figure shows that only one step (the second) is catalysed by a single gene product (that of gene 3). The cell's growth rate of 54.13 is adversely affected if any of its genes are deleted or duplicated (table 6.3.2a), the deletion of gene 3 being lethal, of course. None of the five classes of enzyme of experiment 1 are represented in the cell, all the proto-enzymes of which possess activity towards the first or second (but not both) metabolic steps. The same is true of all other cells sampled at around this point in the experiment (including cells with larger genome sizes and complexities).

As for which of these three genes might be deleted in generating a two-gene cell, table 6.3.2a shows that deleting gene 2 gives a very much higher growth rate (greater than 50) than deleting gene 1 (less than 40). In the present cell, neither deletion is advantageous, but mutational fine-tuning may change that. The gene 1 product has higher activity for all of the steps it shares in common with gene 2, but has no activity for the third metabolic step. Moderate activity for that step is possessed by the proto-enzyme coded by gene

3, however. Furthermore, although this catalyst has no activity for the fifth metabolic step, the gene 1 product does have a reasonable activity for that transformation. It is no suprise, therefore, to find from figure 6.3.2c that the gene pair forming the two-gene cell have activity profiles similar to a <gene 1/gene 3> pair, but with more active enzymes. The product of gene 1 in the cells of both figures has a total of six active steps including step five (but not step three). The other proto-enzyme of the two-gene cell has the same four activities as the product of gene 3 in the earlier cell (the mean value of 5.00 activities per enzyme reflects this 6-4 distribution). The same pattern of activities persists throughout the experiment (figures 6.3.2: d and e). It is now clear that one can do better than the artificially constructed cell of table 6.3.1: proto-enzymes with four moderately high activities are possible in the scenario. The 'sequences' of these proto-enzymes will be presented in a later section.

Genome size and complexity are now firmly fixed. Figure 6.3.1g shows very high levels of mutational death, of which, recalling that deletion, duplication and mutation are equiprobable events, two thirds are presumably due to deletions, and the remaining third to sequence changes eliminating some essential reaction. In the introductory and first experiments, lethal mutations disappeared completely fairly early on. Table 6.3.3b, taking a cell sampled towards the end of the experiment, shows that duplication of either gene is unfavourable, and only minor mutational fine-tuning remains open to the population. Truncation selection, however, seems to have eliminated any possibility of such improvement.

Experiment 2;

Figure 6.3.2

a) Extractable enzyme activities from individual cells sampled when stated.

Profile of the enzyme activities of the fastest cell sampled after:

b) 66,000 cell divisions.
c) 200,000 cell divisions.
d) 500,000 cell divisions.
e) 1,000,000 cell divisions.

(a).

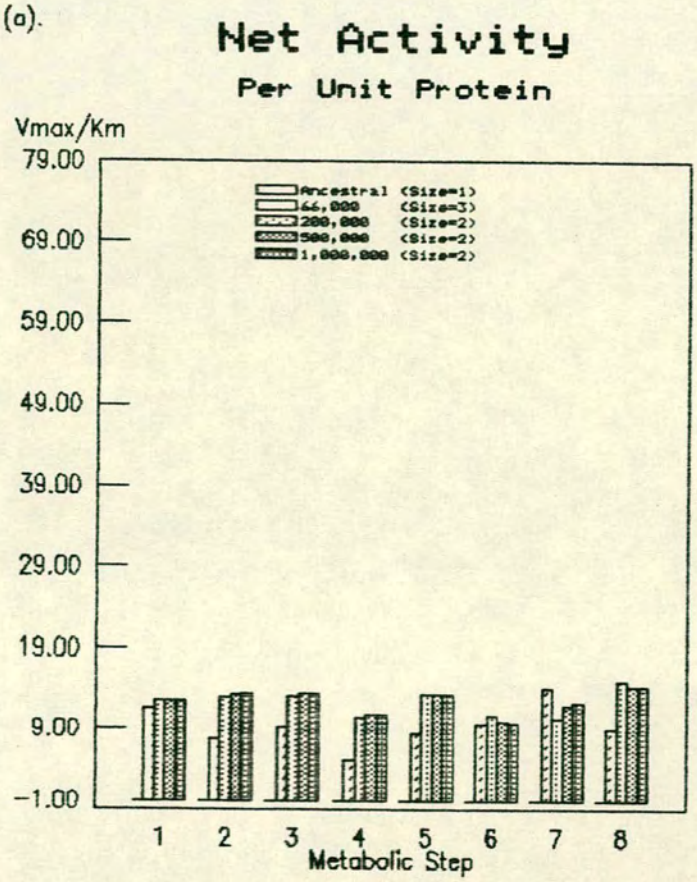
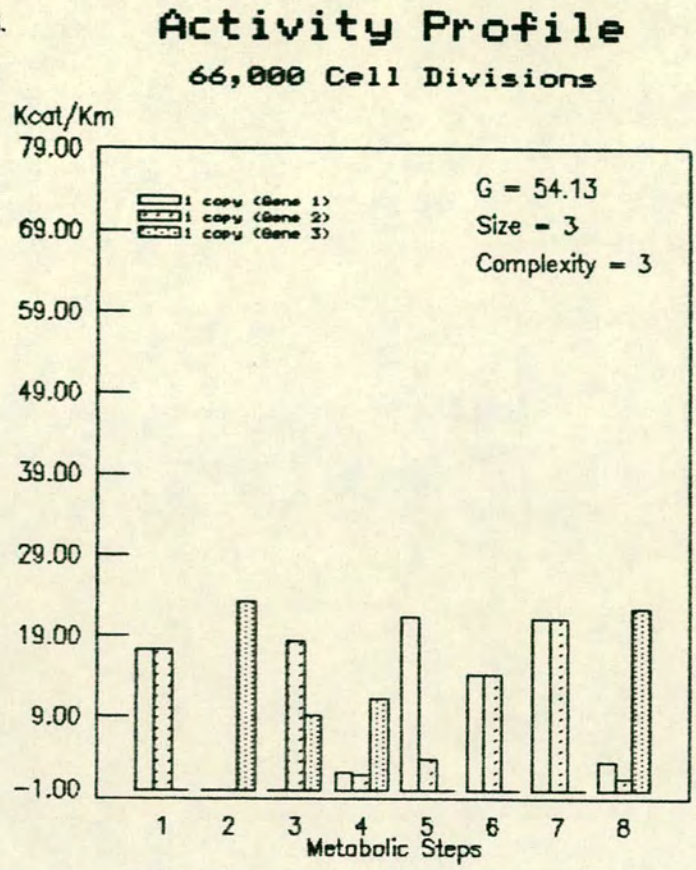
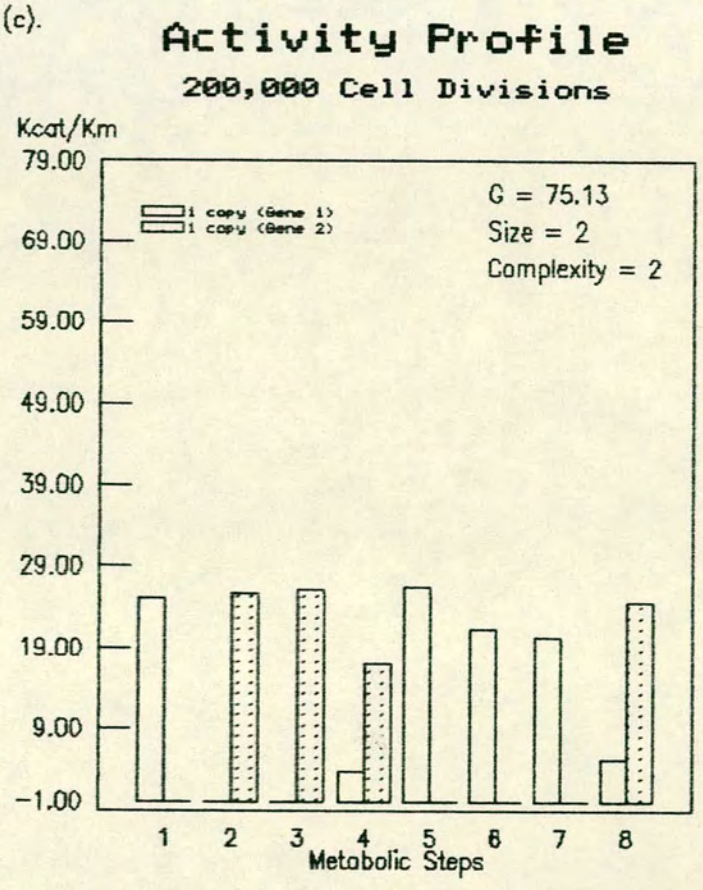


Figure 6.3.2

(b).

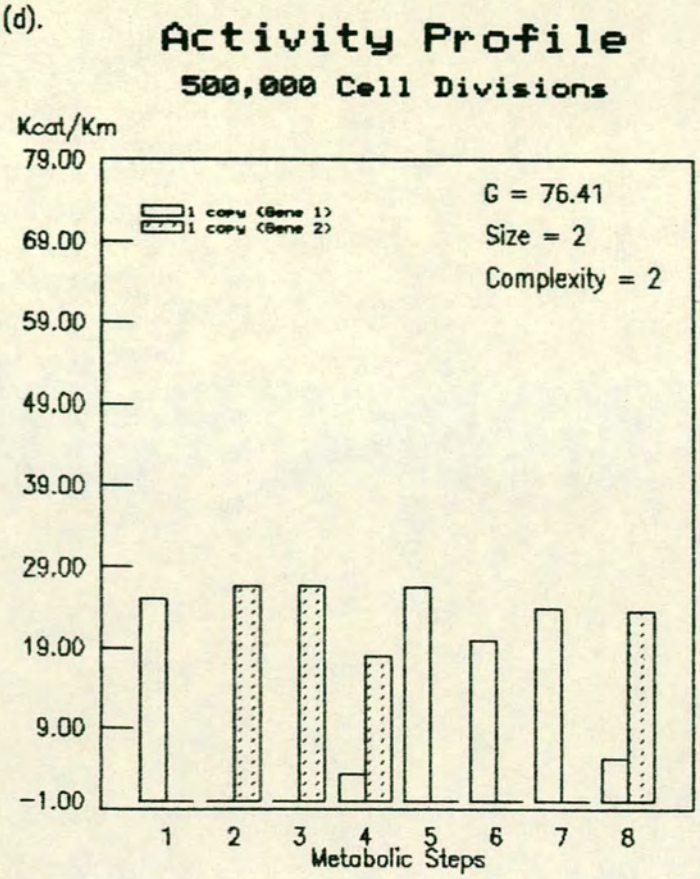

Activity Profile
66,000 Cell Divisions
Kcat/Km
79.00
69.00
59.00
49.00
39.00
29.00
19.00
9.00
-1.00
1 copy (Gene 1)
1 copy (Gene 2)
1 copy (Gene 3)
G = 54.13
Size = 3
Complexity = 3
1 2 3 4 5 6 7 8
Metabolic Steps

(c).

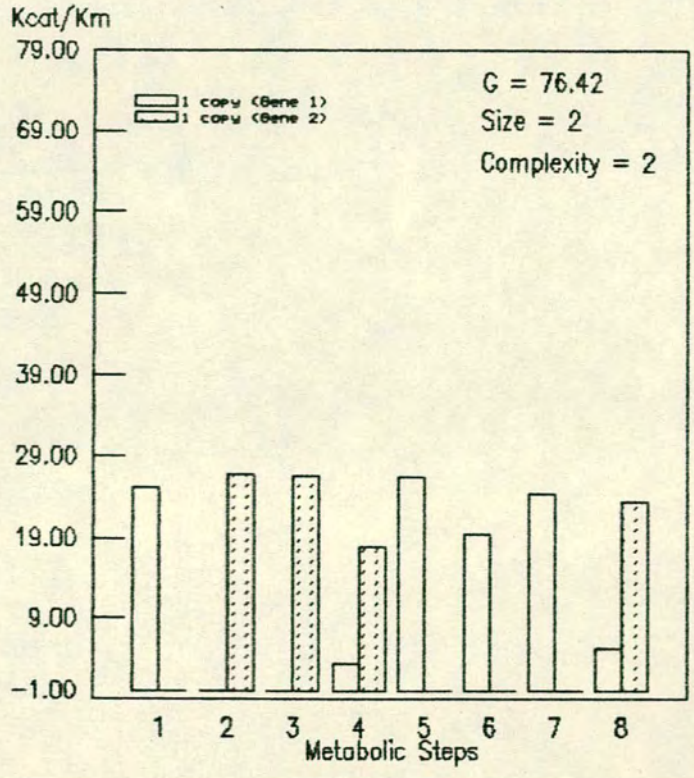

Activity Profile
200,000 Cell Divisions
Kcat/Km
79.00
69.00
59.00
49.00
39.00
29.00
19.00
9.00
-1.00
1 copy (Gene 1)
1 copy (Gene 2)
G = 75.13
Size = 2
Complexity = 2
1 2 3 4 5 6 7 8
Metabolic Steps

gure 6.3.2

(d).


Activity Profile
500,000 Cell Divisions
Kcat/Km
79.00
69.00
59.00
49.00
39.00
29.00
19.00
9.00
-1.00
1 copy (Gene 1)
1 copy (Gene 2)
G = 76.41
Size = 2
Complexity = 2
1 2 3 4 5 6 7 8
Metabolic Steps

(e).


Activity Profile
1,000,000 Cell Divisions
Kcat/Km
79.00
69.00
59.00
49.00
39.00
29.00
19.00
9.00
-1.00
1 copy (Gene 1)
1 copy (Gene 2)
G = 76.42
Size = 2
Complexity = 2
1 2 3 4 5 6 7 8
Metabolic Steps

re 6.3.2

Table 6.3.2

SAMPLED CELL AFTER 66000 CELL DIVISIONS

Genome Size = 3
Genome Complexity = 3
Growth Rate = 54.13153

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	17.40	0.00	0.00	2.30	21.23	14.26	21.14	3.53
2	1	17.37	0.00	18.28	1.93	3.83	14.22	21.05	1.40
3	1	0.00	23.25	9.21	11.29	0.00	0.00	0.00	22.37
V_{max}/K_M		11.59	7.75	9.16	5.18	8.35	9.50	14.07	9.10

EFFECT OF DUPLICATING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	13.04	5.81	6.87	4.46	11.57	10.69	15.83	7.71	50.43066
2	13.03	5.81	11.44	4.36	7.22	10.68	15.81	7.18	50.00237
3	8.69	11.62	9.17	6.70	6.26	7.12	10.55	12.42	53.49960

EFFECT OF DELETING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	8.68	11.62	13.74	6.61	1.92	7.11	10.53	11.89	39.47412
2	8.70	11.62	4.60	6.80	10.61	7.13	10.57	12.95	51.46291
3	17.38	0.00	9.14	2.12	12.53	14.24	21.10	2.47	0.00000

Table 6.3.2 (a)

SAMPLED CELL AFTER 825000 CELL DIVISIONS

Genome Size = 2
Genome Complexity = 2
Growth Rate = 76.41561

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	25.04	0.00	0.00	3.44	26.42	19.51	24.33	5.21
2	1	0.00	26.64	26.63	17.85	0.00	0.00	0.00	23.42
V_{max}/K_M		12.52	13.32	13.32	10.64	13.21	9.75	12.16	14.31

EFFECT OF DUPLICATING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	16.69	8.88	8.88	8.24	17.61	13.00	16.22	11.28	72.41045
2	8.35	17.76	17.76	13.04	8.81	6.50	8.11	17.35	66.01474

EFFECT OF DELETING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	0.00	26.64	26.63	17.85	0.00	0.00	0.00	23.42	0.00000
2	25.04	0.00	0.00	3.44	26.42	19.51	24.33	5.21	0.00000

Table 6.3.2 (b)

Pathway Favouring Multifunctionality - Experiment 3

A rather brief examination of a third repeat of introductory experiment will now be made. The experiment consists, like experiment 2, of two runs of the simulation program, with a truncation factor of 1.00 (figure 6.4.1a). Unlike the second experiment, the present one avoided extinction, but no progress in growth rate at all was made during the second run, which remained at a little above 71 (figure 6.4.1b). At three points during the first run, there were significant falls (nearly 50%) in population size due to truncation selection. In the second run this did not occur.

The real similarity to the second experiment can be seen when the genome size and complexity is examined. The population, early in the first run, adopts a small genome strategy (three single-copy genes) which it retains for the rest of the experiment (figure 6.4.1c). The number of activities per proto-enzyme is high, exceeding the value of 5 obtained in the second experiment (figure 6.4.1d). This is a little surprising in view of the larger genome size, and may help account for the relatively low growth rate (for the individual activities are likely to be lower).

Once all cells in the population have genome sizes and complexities of three, they are very similar to each other. Only one time point is displayed, at one million cell divisions (figure 6.4.1e). This is the fastest cell sampled during the experiment (0.04% faster than the mean, which did not vary throughout the second run) but is not otherwise unusual, mutational fine-tuning of the activities being solely responsible for its better-than-average performance.

The early cell sampled in the previous experiment with three single copy genes has quite a different catalytic profile than the present cell. The cell also differs from the later two-gene cells, in which all but two of the metabolic steps are monocatalysed. The cells of the present experiment have only two such steps (the second and third). There were two metabolic steps in the previous experiment which were catalysed by both proto-enzymes of the cells of the final population (steps 4 and 8), and this applies here also: these two reactions are the only steps catalysed by all three catalysts in the cells. As stated earlier, the transition states for these two steps do not possess any particularly large or small dimensions.

Table 6.4.1 gives details of the sampled cell, and reveals (it comes as no surprise) that there are no favourable single gene duplications or deletions available to it. Only one of the possible deletions is not lethal, as the two monocatalysed reactions are catalysed by different proto-enzymes.

Experiment 3;

Duration: 1.32 million cell divisions.

Sampled: Every 3000 cell divisions.

Truncation factor: 1.00

Figure 6.4.1

a) Population: one bottleneck
b) Mean growth rate of population
c) Mean genome size & complexity
d) Mean activities per proto-enzyme
e) Activity profile of cell sampled
after 1,000,000 cell divisions

(a).

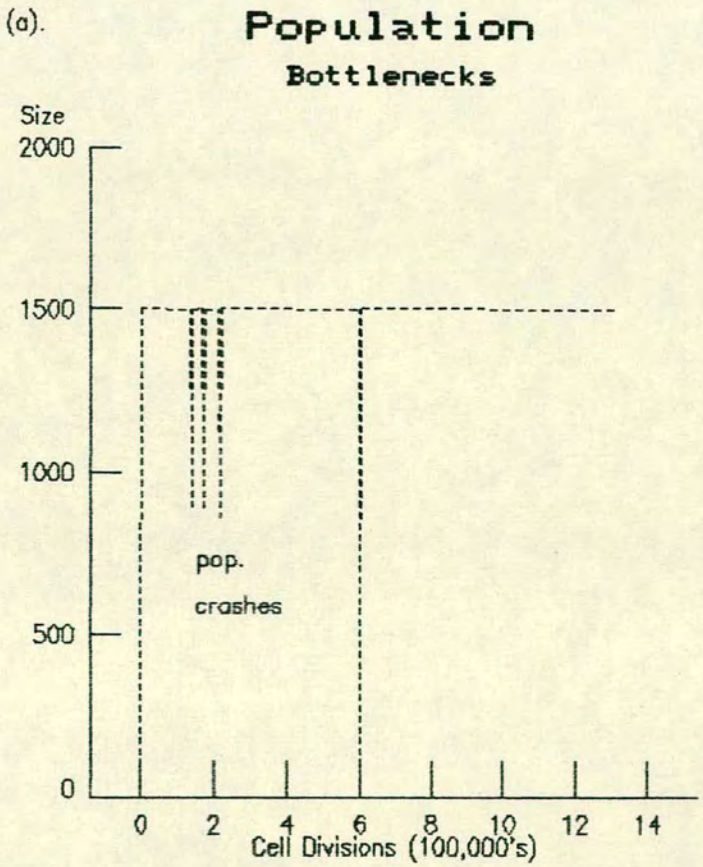
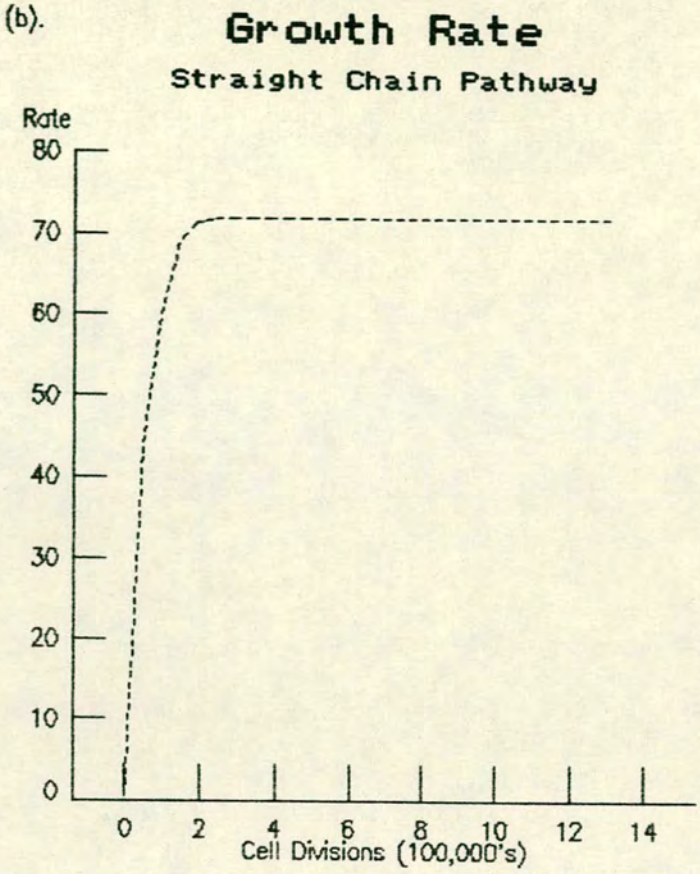
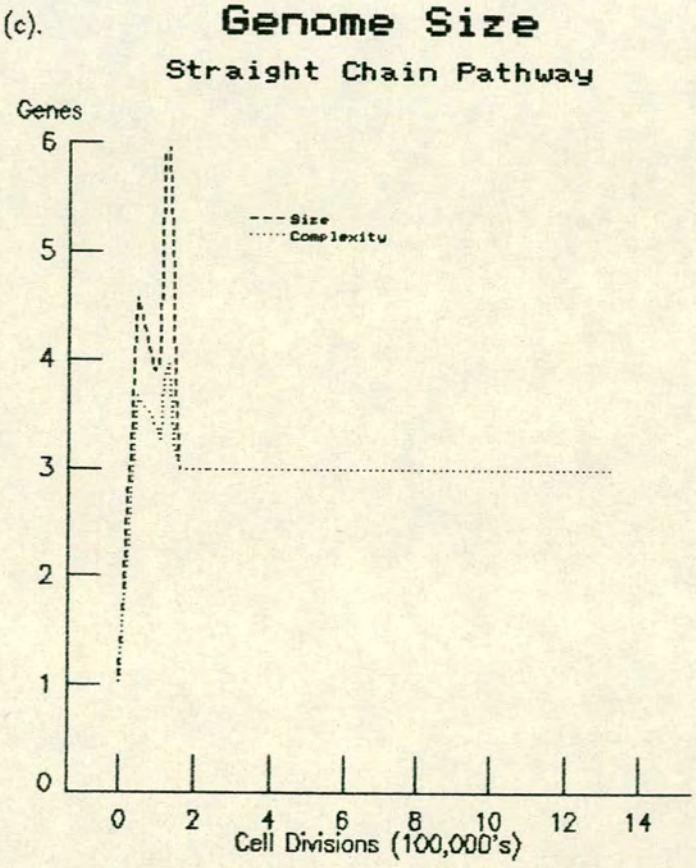


Figure 6.4.1

(b).

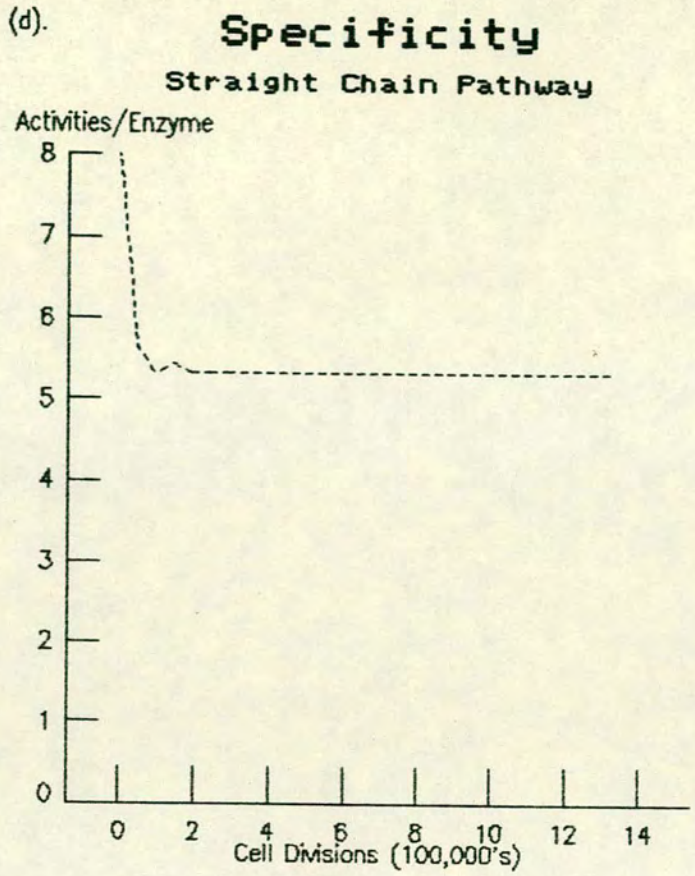

Growth Rate
Straight Chain Pathway
Rate
80
70
60
50
40
30
20
10
0
0
2
4
6
8
10
12
14
Cell Divisions (100,000's)

(c).


Genome Size
Straight Chain Pathway
Genes
6
5
4
3
2
1
0
Size
Complexity
0
2
4
6
8
10
12
14
Cell Divisions (100,000's)

igure 6.4.1

(d).



(e).

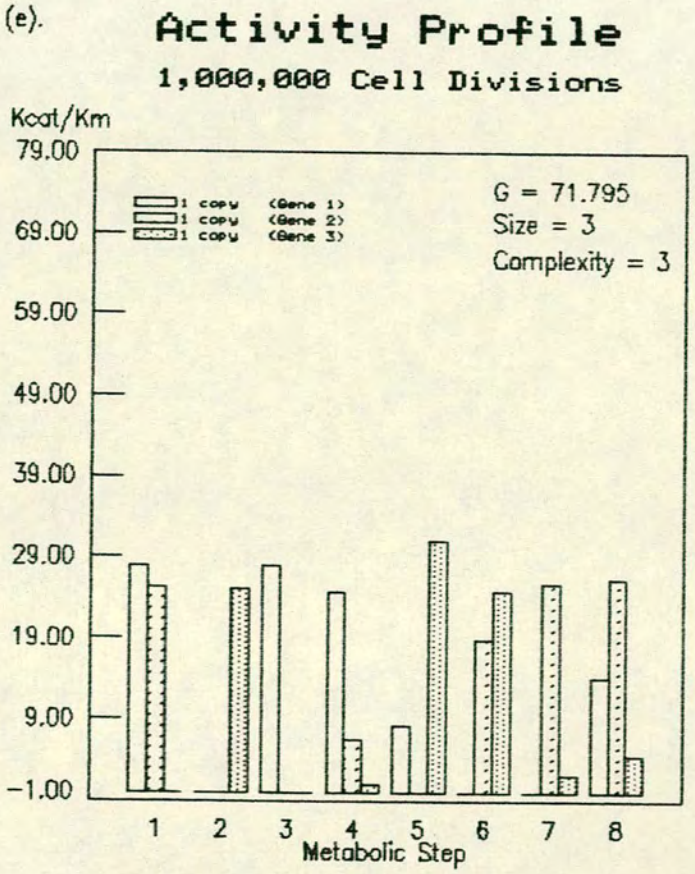


Figure 6.4.1

Table 6.4.1

SAMPLED CELL AFTER 1000000 CELL DIVISIONS

Genome Size = 3
Genome Complexity = 3
Growth Rate = 71.79482

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	27.82	0.00	27.89	24.50	8.29	0.00	0.00	14.22
2	1	25.23	0.00	0.00	6.60	0.00	18.81	25.71	26.18
3	1	0.00	25.12	0.00	1.03	31.02	24.79	2.15	4.54
V_{max}/K_M		17.68	8.37	9.30	10.71	13.10	14.54	9.29	14.98

EFFECT OF DUPLICATING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	20.22	6.28	13.94	14.16	11.90	10.90	6.96	14.79	67.75461
2	19.57	6.28	6.97	9.69	9.83	15.61	13.39	17.78	66.33621
3	13.26	12.56	6.97	8.29	17.58	17.10	7.50	12.37	66.79945

EFFECT OF DELETING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	12.62	12.56	0.00	3.82	15.51	21.80	13.93	15.36	0.00000
2	13.91	12.56	13.94	12.77	19.66	12.40	1.07	9.38	34.00047
3	26.53	0.00	13.94	15.55	4.15	9.41	12.86	20.20	0.00000

Pathway Favouring Multifunctionality - Proto-Enzyme Classes

Experiment 1 led to the notion of proto-enzyme classes, as there were several proto-enzymes in the cell with almost identical activity profiles. Although this is not the case in experiments 2 and 3, the later cells of which contain only single-copy dissimilar genes, the idea of such classes, characterised by the set of reactions towards which they have activity, is a useful one, and the four experiments dealt with so far shall now be examined from the point of view of identifying such classes.

Experiment 2 introduces two new classes of proto-enzyme, which shall be denoted VI (with activities for steps 1, 4, 5, 6, 7, and 8) and VII (catalysing steps 2, 3, 4, and 8). None of these seven classes are represented in the three-gene cells of experiment 3. Instead we have class VIII, which catalyses steps 1, 3, 4, 5, and 8; class IX (activities for 1, 4, 6, 7, and 8); and class X (catalysing steps 2, 4, 5, 6, 7 and 8). It is somewhat arbitrary simply to number the classes as they are encountered, but has the merit for the present purposes that classes are grouped together according to the partners they have evolved with.

We thus have three experiments with no proto-enzyme classes in common at all between them. How many possible proto-enzyme classes are there? It may seem that this is simply a combinatorial question (In how many ways can eight numbers be combined into sets with eight or less elements?) but this ignores the physical constraints on the system. For example, only one of the ten classes so far identified has no activity for steps four and eight (class V).

This is because of a real physical constraint: excluding these reactions excludes most other reactions as well. We shall return to this question shortly, but it is now appropriate to recall that the introductory experiment has not been examined for class content. This is worth doing now.

Table 6.4.2 shows a cell sampled towards the end of the experiment. The cell contains six proto-enzymes, all except one of which are coded for by single-copy genes with different activity profiles. The exception is gene 6, which is present in two copies, and codes for a class V proto-enzyme. The activities are not quite as high as in experiment 1, perhaps because of the comparative youth of the cell. Similarly, gene 2 codes for a class I proto-enzyme, and gene 5 codes for a class II, in both cases with activities a little lower than those of cell 2. No other class present in cell 2 is represented. However, gene 4 codes for a class IX proto-enzyme with similar activities to that of the cell from experiment 3. Neither of the two proto-enzymes of experiment 2 are present. Two thirds of the proto-enzymes of the cell have thus been encountered in other experiments. This gives two additional classes to add to the list: class XI (catalysing steps 3, 4, 5, and 8) and class XII (with activity towards 3, 4, 5, 6, 7, and 8).

The low genome size again suggests that the optimum ratios of the proto-enzymes to each other have not been reached (but recall that we have seen that at least three of them can be mutationally improved), and table 6.4.2 does indeed show that there are no favourable duplications or deletions in the cell. In the discussion of experiment 1, a simple procedure of multiplying the number of copies of each gene in the genome by five to test how close to

optimum the cells were was tried, and this shall now be adopted for each of the other three experiments considered to date (table 6.4.3). Not surprisingly, favourable gene copy changes exist for all three populations when so treated. The removal of the constraint in the experimental design that only single duplications or deletions are allowed would clearly allow each gene set optimum to be more closely approached. A simple change to the program, retaining the constraint while allowing whole genome duplication (polyploidy) would possibly be sufficient to allow the cells of these experiments to shed some of the shackles they wear.

Table 6.4.2

CELL SAMPLED AT END OF INTRODUCTORY EXPERIMENT

Genome Size = 7
Genome Complexity = 6
Growth Rate = 80.08839

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	0.00	27.89	25.58	32.18	0.00	0.00	14.96
2	1	0.00	0.00	28.61	41.45	0.00	0.00	0.00	37.76
3	1	0.00	0.00	25.28	3.28	11.60	34.89	23.96	2.37
4	1	25.59	0.00	0.00	6.93	0.00	20.39	23.04	26.24
5	1	0.00	0.00	0.00	3.95	49.67	0.00	32.83	6.59
6	2	44.14	44.58	0.00	0.00	0.00	22.13	0.00	0.00
V_{max}/K_M		16.27	12.74	11.68	11.60	13.35	14.22	11.40	12.56

EFFECT OF DUPLICATING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	14.23	11.15	13.71	13.35	15.70	12.44	9.98	12.86	79.38339
2	14.23	11.15	13.80	15.33	11.68	12.44	9.98	15.71	79.68079
3	14.23	11.15	13.38	10.56	13.13	16.81	12.97	11.29	79.27125
4	17.43	11.15	10.22	11.02	11.68	14.99	12.86	14.27	78.63660
5	14.23	11.15	10.22	10.64	17.89	12.44	14.08	11.81	77.67985
6	19.75	16.72	10.22	10.15	11.68	15.21	9.98	10.99	77.00033

EFFECT OF DELETING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	18.98	14.86	8.98	9.27	10.21	16.59	13.31	12.16	76.35920
2	18.98	14.86	8.86	6.62	15.58	16.59	13.31	8.36	71.11819
3	18.98	14.86	9.42	12.98	13.64	10.78	9.31	14.26	77.30362
4	14.71	14.86	13.63	12.38	15.58	13.19	9.47	10.28	79.07128
5	18.98	14.86	13.63	12.87	7.30	16.59	7.83	13.55	74.65119
6	11.62	7.43	13.63	13.53	15.58	12.90	13.31	14.65	76.56971

Table 6.4.3

CELL SAMPLED AT END OF INTRODUCTORY EXPERIMENT
WITH ALL GENE DOSAGES INCREASED FIVE-FOLD

Genome Size = 35
Genome Complexity = 6
Growth Rate = 80.08839

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	0.00	0.00	27.89	25.58	32.18	0.00	0.00	14.96
2	5	0.00	0.00	28.61	41.45	0.00	0.00	0.00	37.76
3	5	0.00	0.00	25.28	3.28	11.60	34.89	23.96	2.37
4	5	25.59	0.00	0.00	6.93	0.00	20.39	23.04	26.24
5	5	0.00	0.00	0.00	3.95	49.67	0.00	32.83	6.59
6	10	44.14	44.58	0.00	0.00	0.00	22.13	0.00	0.00
V_{max}/K_M		16.27	12.74	11.68	11.60	13.35	14.22	11.40	12.56

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
36	6	Duplicate	2	80.42508	1 3;	4 6
37	6	Duplicate	2	80.50476	3;	4 6
38	6	Duplicate	3	80.58588		1 4
37	6	Delete	4	80.66072		
36	6	Delete	1	80.72559		
35	6	Delete	4	80.77748		
35	6	None	0	80.77748		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 35
Genome Complexity = 6
Growth Rate = 80.77748

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	4	0.00	0.00	27.89	25.58	32.18	0.00	0.00	14.96
2	7	0.00	0.00	28.61	41.45	0.00	0.00	0.00	37.76
3	6	0.00	0.00	25.28	3.28	11.60	34.89	23.96	2.37
4	3	25.59	0.00	0.00	6.93	0.00	20.39	23.04	26.24
5	5	0.00	0.00	0.00	3.95	49.67	0.00	32.83	6.59
6	10	44.14	44.58	0.00	0.00	0.00	22.13	0.00	0.00
V_{max}/K_M		14.80	12.74	13.24	12.93	12.76	14.05	10.77	12.86

CELL SAMPLED AFTER 825,000 CELL DIVISIONS IN EXPERIMENT 2
WITH ALL GENE DOSAGES INCREASED FIVE-FOLD

Genome Size = 10
Genome Complexity = 2
Growth Rate = 76.41575

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	25.04	0.00	0.00	3.44	26.42	19.51	24.33	5.21
2	5	0.00	26.64	26.63	17.85	0.00	0.00	0.00	23.42
V_{max}/K_M		12.52	13.32	13.32	10.64	13.21	9.75	12.16	14.31

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
11	2	Duplicate	1	76.75499		2
11	2	None	0	76.75499		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 11
Genome Complexity = 2
Growth Rate = 76.75499

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	6	25.04	0.00	0.00	3.44	26.42	19.51	24.33	5.21
2	5	0.00	26.64	26.63	17.85	0.00	0.00	0.00	23.42
V_{max}/K_M		13.66	12.11	12.11	9.99	14.41	10.64	13.27	13.49

Table 6.4.3

CELL SAMPLED AFTER 1,000,000 CELL DIVISIONS OF EXPERIMENT 3
WITH ALL GENE DOSAGES INCREASED FIVE-FOLD

Genome Size = 15
Genome Complexity = 3
Growth Rate = 71.79482

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	27.82	0.00	27.89	24.50	8.29	0.00	0.00	14.22
2	5	25.23	0.00	0.00	6.60	0.00	18.81	25.71	26.18
3	5	0.00	25.12	0.00	1.03	31.02	24.79	2.15	4.54
V_{max}/K_M		17.68	8.37	9.30	10.71	13.10	14.54	9.29	14.98

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
14	3	Delete	2	71.80692		
14	3	None	0	71.80692		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 14
Genome Complexity = 3
Growth Rate = 71.80692

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	27.82	0.00	27.89	24.50	8.29	0.00	0.00	14.22
2	4	25.23	0.00	0.00	6.60	0.00	18.81	25.71	26.18
3	5	0.00	25.12	0.00	1.03	31.02	24.79	2.15	4.54
V_{max}/K_M		17.14	8.97	9.96	11.01	14.04	14.23	8.11	14.18

Proto-Enzyme Classes: Additional Experiments

The introductory experiment differed from experiments 1, 2, and 3 in a number of ways. Firstly, the mutation probability of any given gene was independent of its dosage (i.e. a gene present in a single copy was as likely to be duplicated, deleted or mutated as one present in ten copies). Secondly, the truncation selection for the experiment was weaker. Thirdly, the population size was much larger.

In experiments 1, 2, and 3, the population was stored in an array (a set of contiguous locations in memory each of which can be directly accessed). This method of storage has some advantages and some disadvantages. The major disadvantage is that the array is stored on the main program stack. On the Data General/Rolm Ada implementation, which executes on a machine with virtual memory (disc space is treated as though it were main memory), the space for this stack is included by the linker in the final executable file, so that for a population size of 1000 the file is over two and half megabytes in size (each cell requires some two kilobytes of storage, to provide for its forty possible genes, with the catalytic coefficients for each of them with respect to each of the eight reactions stored as eight-byte floating point numbers). Population sizes of 3500 or 4000 are not really practical in the face of these requirements. (Of course, all the coefficients could be recalculated at every cell division, but the execution time of the program is already very high, and storing the value means that such recalculation is only required when a gene mutates, and then only for that single gene). A related disadvantage, if the array becomes very large, concerns the paging of sections of this array from

disc into memory, particularly problematic if there is only a low correlation between the points of the array accessed in successive operations (as, for example, when sampling).

In the introductory experiment the population of cells was kept in dispersed, so-called 'nameless', storage, on the heap. This makes life simpler for the run-time system in some ways, as there is no preference for contiguous blocks of memory to be used, and space can be scavenged wherever available. For the user (in this case, me) the inconvenience of extremely large executable files is avoided while allowing populations of quite substantial size (5000 or so, though as size is increased towards this level, the share of cpu is reduced because of memory management overheads) to be maintained.

However, the ability to directly access each cell is a very considerable advantage in the present experiment. Firstly, it allows sampling of arbitrary cells in the population, and so provides, secondly, for 'random' deaths. Both of these are prohibitively expensive in the implementation of the introductory experiment. In that experiment, the sampling process simply samples cells as they present for division. The 'random' deaths are not random at all, but involve killing cells most distantly removed in simulation time from cell division (this is easy to implement because they are located at the 'end' of a circular doubly linked list which can alway be directly accessed via the pointer indicating the front, where the next cell due for division is located).

In a population with considerable variation these cells will tend to be the slowest ones, and 'random' deaths in response to overpopulation do, in fact,

have a considerable selective component. In a population with little variation in growth rate, however, such cells will be the cells which have most recently divided, and progress of the simulation (already difficult in the absence of such variation) will be impeded. The high truncation factor of the introductory experiment is to attempt to maintain higher variation in growth rates and avoid the anti-progressive consequences of killing cells on the basis of the relative interval to their next scheduled cell division. It was this difficulty, more than any other, which led to the bulk of the work being performed using the array implementation.

Nevertheless, a number of additional simulations were performed using the dispersed storage implementation. The progress of the two largest of these experiments are presented in figure 6.4.2, and cells drawn from the final populations are shown in figures 6.4.3 and 6.4.4. In both experiments fairly small genome sizes were obtained with some variation in size and complexity (presumably because of the weak selection, as growth rates are also more variable). The adaptive solutions reached were again based on a small number (four in one case, five in the other) of proto-enzyme classes possessed in common by the cells of the populations concerned (but see below) and persisting for long periods of simulation time. Growth rates were still improving by mutational fine-tuning when the experiments were laid to rest.

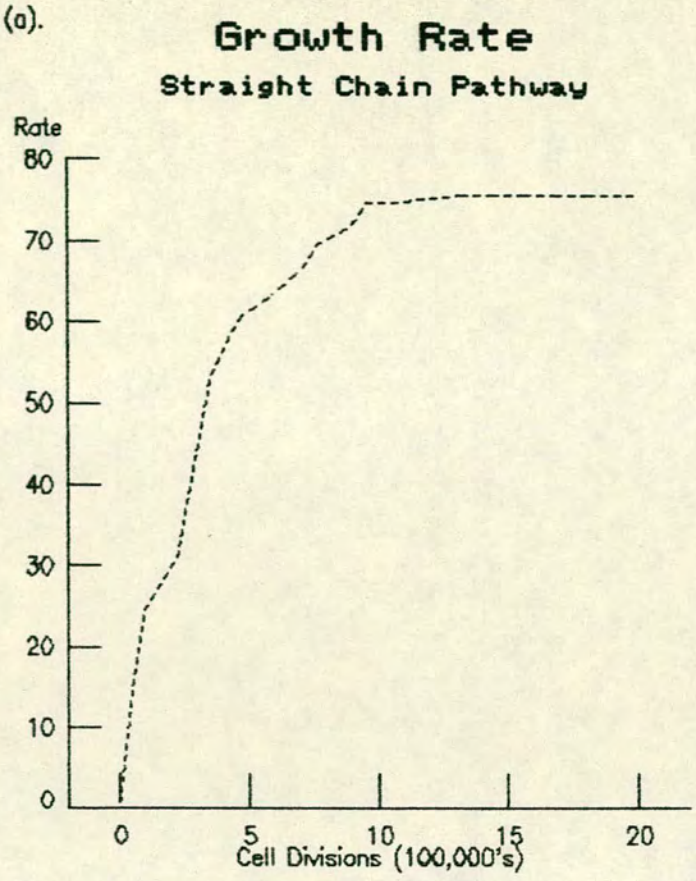
Particularly interesting in A2, the second of the two experiments shown, is that two different five-class solutions were coexisting at the end of the experiment, although the period of coexistence had not been long, and would probably have led to the elimination of one of them if the simulation had been

extended. The cell types are identical with respect to four of the five proto-enzyme classes, but differ in the other (that coded by gene 3 of both types of cell). In one of the cell types, gene 3 codes for a proto-enzyme that catalyses steps 1, 4, 6, and 7; in the other activity is restricted to steps 1, 6 and 7. Cells of the former kind seem to grow more slowly. Only one of the twelve classes so far encountered (class III), are represented in this experiment (similarly for experiment A1, where the class concerned is XI).

Although one of the five classes in each cell type is different, the relative allocation to each class is the same and cannot be improved by duplication or deletion. However, increasing the dosage of each gene five-fold allows a series of a few favourable duplications and deletions (table 6.4.4, parts a and b), as has been the case in all the experiments. It is probably not surprising to now find that the relative allocation to the different classes does differ between the cell types.

One final point on this 'class polymorphism' of experiment A2. Examination of the cells in the table reveals that there are differences in the quality of the proto-enzymes with which they are endowed for all of the classes they share in common. To determine what effect these differences have on their relative performance, two composite cells were constructed using the more active proto-enzyme of either cell to represent the common classes. These are easy to identify for all except the product of gene 2, where it seems possible that the relative performance of the alternatives might be different in the two cell types. In fact, one of the genes is better in both cell types, as determined by the simple method of trying first one variant and then the other. Table

(a).



(b).

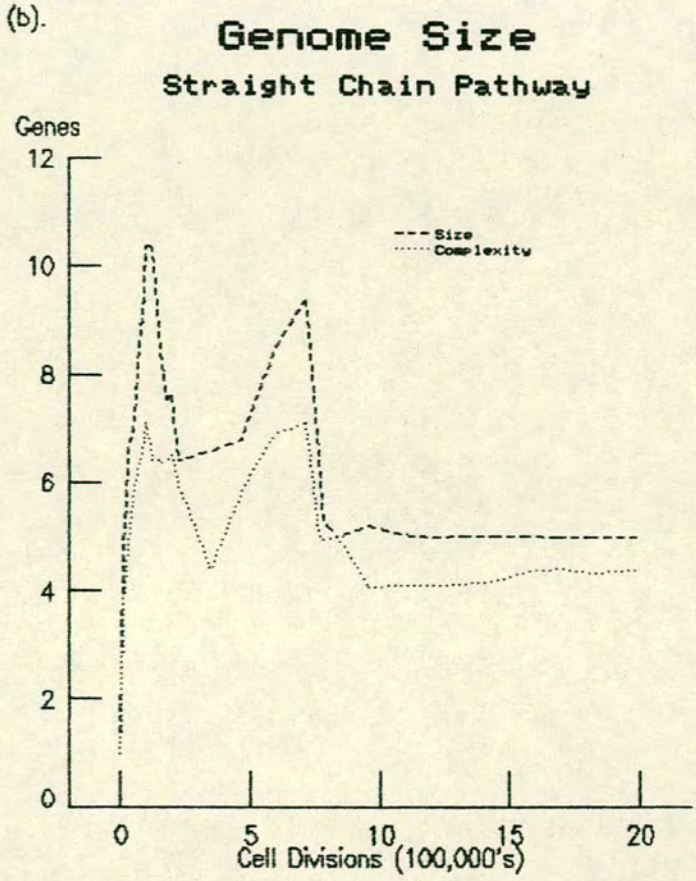
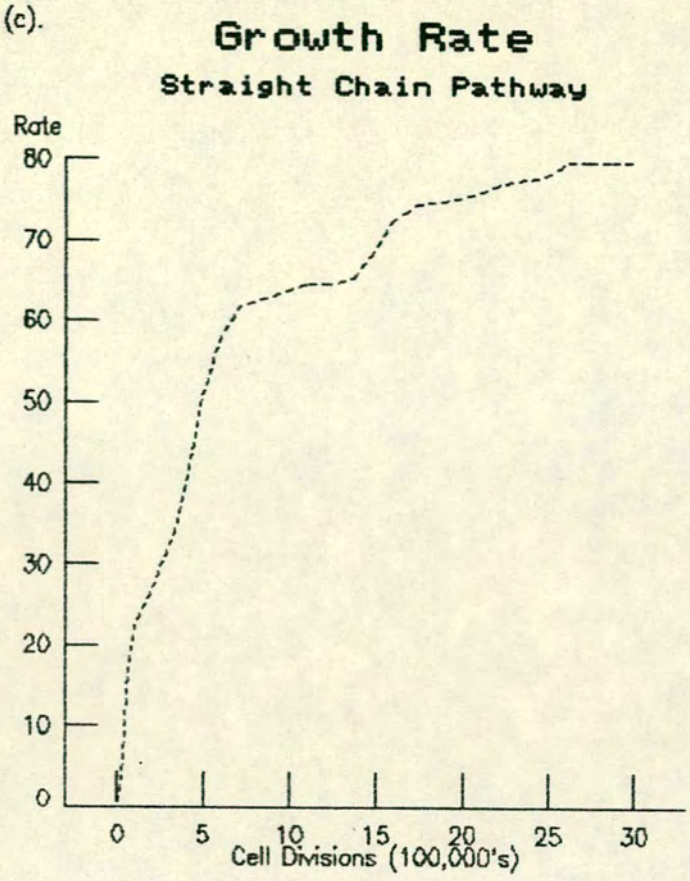


Figure 6.4.2

(c).



(d).

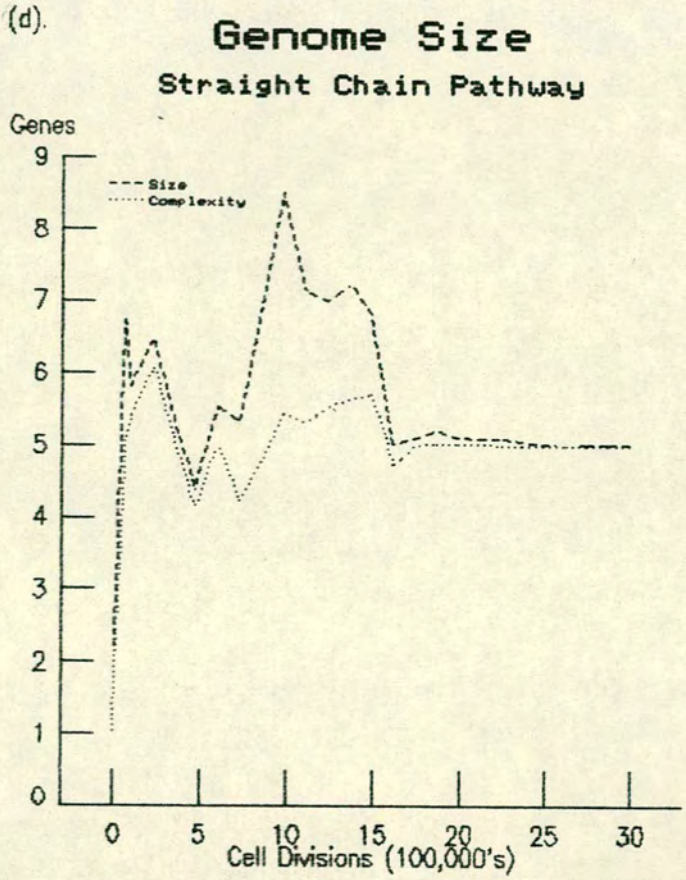


Figure 6.4.2 *Experiment A2* 254

Experiment A1

Duration: 2 million cell divisions.

Truncation factor: 1.02

Population size: 3500

***Figure* 6.4.3**

a) Slowest growing cell sampled at 2,000,000 cell divisions.

b) Fastest growing cell sampled at 2,000,000 cell divisions.

Experiment A2

Duration: 3 million cell divisions.

Truncation factor: 1.08

Population size: 3500

Figure* 6.4.4 – *Polymorphism

a) Fastest–growing cell sampled 2,800,000 cell divisions.

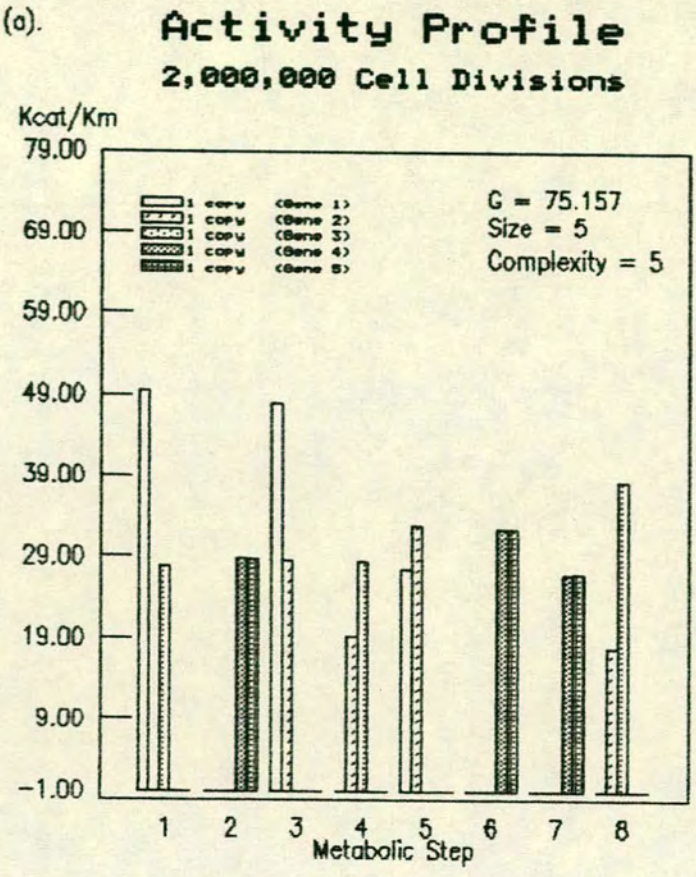
b) Fastest growing cell sampled 2,880,000 cell divisions.

c) Slowest–growing cell sampled 2,960,000 cell divisions.

d) Fastest–growing cell sampled 2,960,000 cell divisions.

igure 6.4.3/4

(a).



(b).

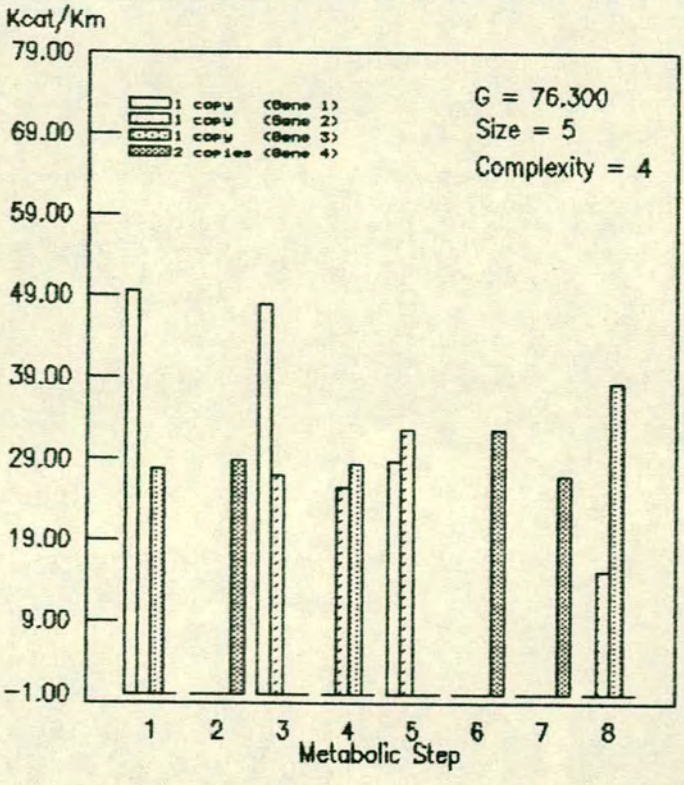
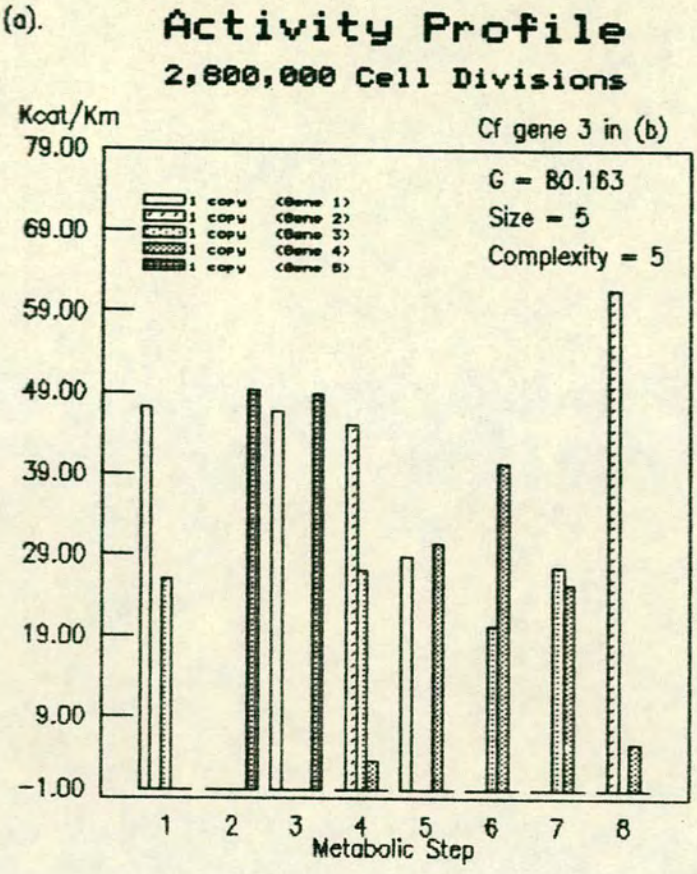


Figure 6.4.3

(a).



(b).

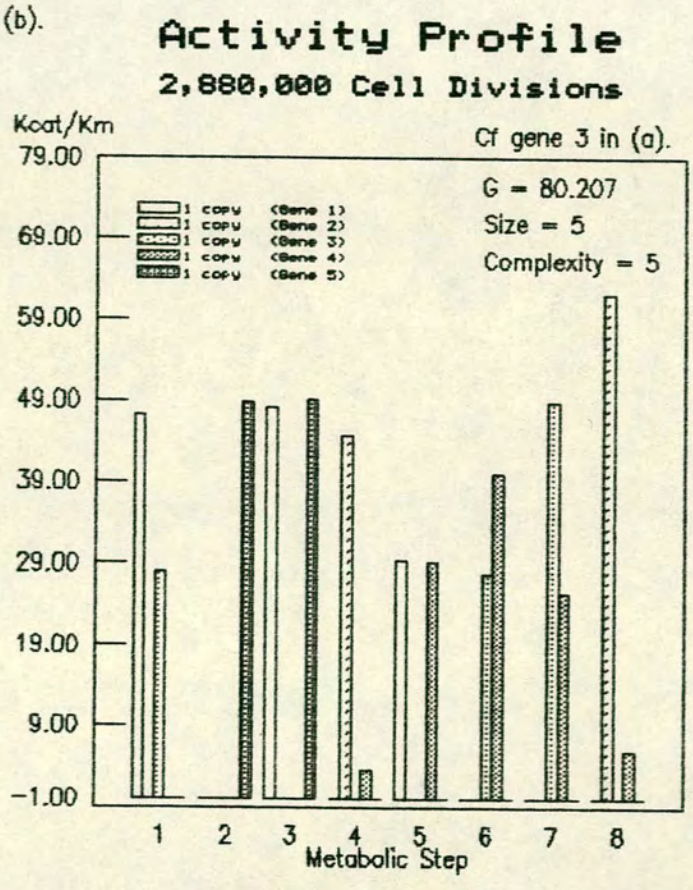
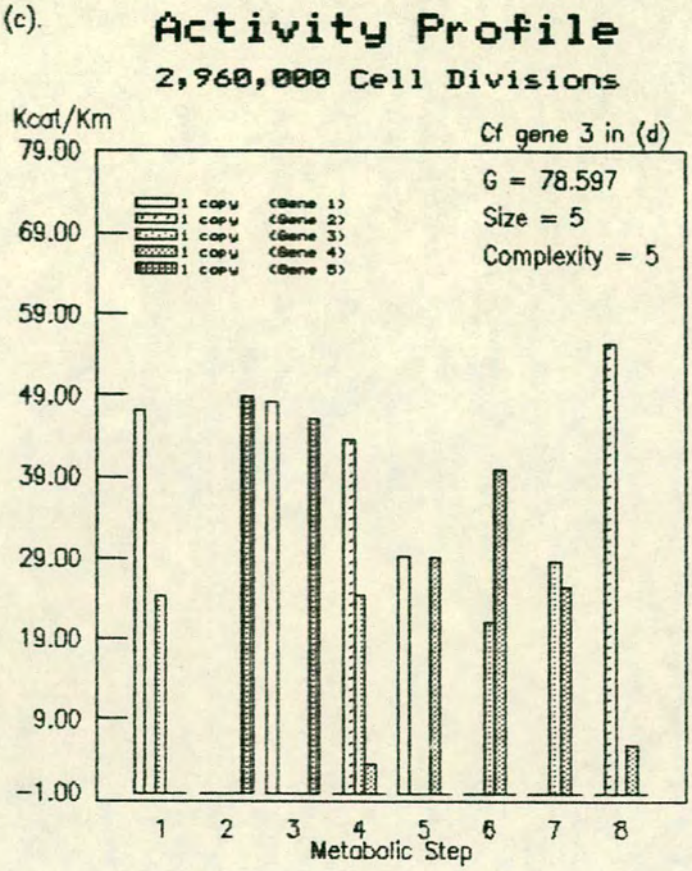


Figure 6.4.4 *Experiment A2*

(c).



(d).

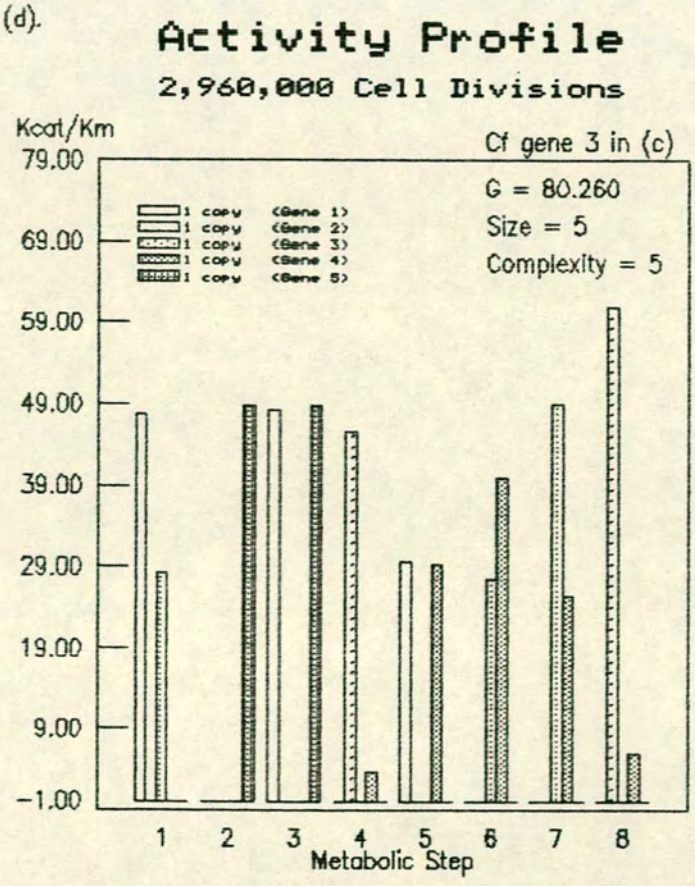


Figure 6.4.4

Table 6.4.4

(a)

FASTEST CELL SAMPLED AFTER 2.8m CELL DIVISIONS OF
EXPERIMENT A2 WITH ALL GENE DOSAGES INCREASED FIVE-FOLD

Genome Size = 25
Genome Complexity = 5
Growth Rate = 80.16285

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	47.15	0.00	46.76	0.00	28.93	0.00	0.00	0.00
2	5	0.00	0.00	0.00	45.02	0.00	0.00	0.00	61.73
3	5	26.05	0.00	0.00	27.01	0.00	20.21	27.50	0.00
4	5	0.00	0.00	0.00	3.63	30.47	40.26	25.40	5.73
5	5	0.00	49.27	48.76	0.00	0.00	0.00	0.00	0.00
V_{max}/K_M		14.64	9.85	19.10	15.13	11.88	12.09	10.58	13.49

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
26	5	Duplicate	4	80.66159		1 2
27	5	Duplicate	4	80.78246		1 3
28	5	Duplicate	5	80.97239		1 3
29	5	Duplicate	4	81.02667		
29	5	None	0	81.02667		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 29
Genome Complexity = 5
Growth Rate = 81.02667

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	47.15	0.00	46.76	0.00	28.93	0.00	0.00	0.00
2	5	0.00	0.00	0.00	45.02	0.00	0.00	0.00	61.73
3	5	26.05	0.00	0.00	27.01	0.00	20.21	27.50	0.00
4	8	0.00	0.00	0.00	3.63	30.47	40.26	25.40	5.73
5	6	0.00	49.27	48.76	0.00	0.00	0.00	0.00	0.00
V_{max}/K_M		12.62	10.19	18.15	13.42	13.39	14.59	11.75	12.23

(b)

FASTEST CELL SAMPLED AT END OF EXPERIMENT A2
WITH ALL GENE DOSAGES INCREASED FIVE-FOLD

Genome Size = 25
Genome Complexity = 5
Growth Rate = 80.25934

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	47.69	0.00	48.23	0.00	29.57	0.00	0.00	0.00
2	5	0.00	0.00	0.00	45.45	0.00	0.00	0.00	60.84
3	5	28.18	0.00	0.00	0.00	0.00	27.37	49.03	0.00
4	5	0.00	0.00	0.00	3.66	29.24	39.95	25.35	5.96
5	5	0.00	48.86	48.75	0.00	0.00	0.00	0.00	0.00
V_{max}/K_M		15.17	9.77	19.40	9.82	11.76	13.46	14.88	13.36

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
24	5	Delete	3	80.74295	2;	1
25	5	Duplicate	2	80.95702		1 3
26	5	Duplicate	4	81.07030		
26	5	None	0	81.07030		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 26
Genome Complexity = 5
Growth Rate = 81.07030

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	47.69	0.00	48.23	0.00	29.57	0.00	0.00	0.00
2	6	0.00	0.00	0.00	45.45	0.00	0.00	0.00	60.84
3	4	28.18	0.00	0.00	0.00	0.00	27.37	49.03	0.00
4	6	0.00	0.00	0.00	3.66	29.24	39.95	25.35	5.96
5	5	0.00	48.86	48.75	0.00	0.00	0.00	0.00	0.00
V_{max}/K_M		13.51	9.40	18.65	11.33	12.43	13.43	13.39	15.42

Table 6.4.4

(c)

CELL CONSTRUCTED USING 'BEST' GENES FROM CELLS IN (a) AND (b)
WITH GENE 3 FROM (a).

Genome Size = 25
Genome Complexity = 5
Growth Rate = 80.33556

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	47.69	0.00	48.23	0.00	29.57	0.00	0.00	0.00
2	5	0.00	0.00	0.00	45.02	0.00	0.00	0.00	61.73
3	5	26.05	0.00	0.00	27.01	0.00	20.21	27.50	0.00
4	5	0.00	0.00	0.00	3.63	30.47	40.26	25.40	5.73
5	5	0.00	48.86	48.75	0.00	0.00	0.00	0.00	0.00
V_{max}/K_M		14.75	9.77	19.40	15.13	12.01	12.09	10.58	13.49

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
26	5	Duplicate	4	80.82203		1 2
27	5	Duplicate	4	80.93269		1 3
28	5	Duplicate	5	81.12892		1 3
29	5	Duplicate	4	81.17487		
29	5	None	0	81.17487		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 29
Genome Complexity = 5
Growth Rate = 81.17487

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	47.69	0.00	48.23	0.00	29.57	0.00	0.00	0.00
2	5	0.00	0.00	0.00	45.02	0.00	0.00	0.00	61.73
3	5	26.05	0.00	0.00	27.01	0.00	20.21	27.50	0.00
4	8	0.00	0.00	0.00	3.63	30.47	40.26	25.40	5.73
5	6	0.00	48.86	48.75	0.00	0.00	0.00	0.00	0.00
V_{max}/K_M		12.71	10.11	18.40	13.42	13.50	14.59	11.75	12.23

(d)

CELL FORMED FROM THE 'BEST' GENES OF (a) AND (b)
WITH GENE 3 FROM (b)

Genome Size = 25
Genome Complexity = 5
Growth Rate = 80.61442

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	47.69	0.00	48.23	0.00	29.57	0.00	0.00	0.00
2	5	0.00	0.00	0.00	45.02	0.00	0.00	0.00	61.73
3	5	28.18	0.00	0.00	0.00	0.00	27.37	49.03	0.00
4	5	0.00	0.00	0.00	3.63	30.47	40.26	25.40	5.73
5	5	0.00	49.27	48.76	0.00	0.00	0.00	0.00	0.00
V_{max}/K_M		15.17	9.85	19.40	9.73	12.01	13.53	14.89	13.49

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
26	5	Duplicate	2	81.11226		1 3
25	5	Delete	3	81.33853		
26	5	Duplicate	4	81.45734		
26	5	None	0	81.45734		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 26
Genome Complexity = 5
Growth Rate = 81.45734

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	47.69	0.00	48.23	0.00	29.57	0.00	0.00	0.00
2	6	0.00	0.00	0.00	45.02	0.00	0.00	0.00	61.73
3	4	28.18	0.00	0.00	0.00	0.00	27.37	49.03	0.00
4	6	0.00	0.00	0.00	3.63	30.47	40.26	25.40	5.73
5	5	0.00	49.27	48.76	0.00	0.00	0.00	0.00	0.00
V_{max}/K_M		13.51	9.47	18.65	11.23	12.72	13.50	13.41	15.57

able 6.4.4

6.4.4 (c and d) reveals that the growth rates of both cell types are improved by this procedure (of creating a composite cell using the best genes), but that the relative evaluation of the two strategies remains unchanged (the gene 3 variant catalysing three steps gives better growth than the one catalysing four). Rather interestingly, the relative allocation of protein in these composite cells converges to the same point as in the originals, but if parts (b) and (d) of the table are compared, it will be seen that the sequence of duplications and deletions has been altered.

(It must be bourne in mind that the procedure used to generate these tables only looks at the best of the gene copy number changes. Others, yielding higher growth rates than the parent but less than the event actually chosen, are ignored but could in principle lead to better solutions if they were pursued. The few trials I tried when preparing chapter 2, in which the same approach is also used in the discussion on optimum allocations, did not produce any superior final points, but the question has not been investigated in any real sense. The point of the exercise is to show that the allocation can be improved if larger numbers of genes are available).

Both of the experiments introduce new proto-enzyme classes in new combinations and the list of twelve gathered so far is extended in table 6.4.5 below. One common feature of the strategies adopted in experiments A1 and A2 which sets them a little aside from the previous ones we have seen is that the number of activities per proto-enzyme is rather lower. In both cases, this has been accompanied (cause or effect?) by the evolution of specialist enzymes dealing with reactions four and eight. The growth rates do not reach the

heights obtained in experiment 1 (even when the five-fold expanded cells are compared with the unexpanded cells of that experiment) and the question of the 'best solution' remains an open one.

It is a surprising and pleasing quality of the present scenario that there are such a diversity of possible adaptations to it, and that these adaptations involve making a commitment from which it is usually (but not always as experiment A2 shows) impossible to escape. The local topography of the adaptive landscape is all-important after initial fortuitous duplications and mutations have thrown the population somewhere upon it.

Table 6.4.5

Experiments with Pathway Favouring Multifunctionality

Allocation of Protein

Class	*Activities (Steps)*	*Experiment*													
		Intro	*x5*	*One*	*x5*	*Two*	*x5*	*Three*	*x5*	*A1*	*x5*	*A2*	*x5[a]*	*x5[b]*
		(7)	*(35)*	*(23)*	*(116)*	*(2)*	*(11)*	*(3)*	*(14)*	*(5)*	*(25)*	*(5)*	*(29)*	*(26)*
I	3 4 8	1	7	7	35	0	0	0	0	0	0	0	0	0
II	4 5 7 8	1	5	1	5	0	0	0	0	0	0	0	0	0
III	4 5 6 7 8	0	0	4	22	0	0	0	0	0	0	1	8	6
IV	3 4 5 7 8	0	0	4	19	0	0	0	0	0	0	0	0	0
V	1 2 6	2	10	7	35	0	0	0	0	0	0	0	0	0
VI	1 4 5 6 7 8	0	0	0	0	1	6	0	0	0	0	0	0	0
VII	2 3 4 8	0	0	0	0	1	4	0	0	0	0	0	0	0
VIII	1 3 4 5 8	0	0	0	0	0	0	1	5	0	0	0	0	0
IX	1 4 6 7 8	1	3	0	0	0	0	1	4	0	0	0	0	0
X	2 4 5 6 7 8	0	0	0	0	0	0	1	5	0	0	0	0	0
XI	3 4 5 8	1	4	0	0	0	0	0	0	1	6	0	0	0
XII	3 4 5 6 7 8	1	7	0	0	0	0	0	0	0	0	0	0	0
XIII	1 4 8	0	0	0	0	0	0	0	0	1	5	0	0	0
XIV	1 3 5	0	0	0	0	0	0	0	0	1	4	0	0	0
XV	2 6 7	0	0	0	0	0	0	0	0	2	10	1	5	5
XVI	4 8	0	0	0	0	0	0	0	0	0	0	1	5	6
XVII	1 4 6 7	0	0	0	0	0	0	0	0	0	0	1[a]	5	0
XVIII	1 6 7	0	0	0	0	0	0	0	0	0	0	1[b]	0	4
XIX	2 3	0	0	0	0	0	0	0	0	0	0	1	6	5

NB, The superscripts a *and* b *refer to the two cell types present at the end of experiment A2.*

Perfect Enzymes?

The model of enzyme catalysis used in these experiments views the interaction between active site and transition state in terms of the 'box-brick' theory. The active site is complementary to the transition state if it binds it perfectly: that is, if the Lennard-Jones function for each of the three axes of interaction is at its minimum point (maximum negative ΔE).

In the four experiments we have so far discussed, the pathway (set of transition states) allows the existence of monofunctional enzymes. Such proto-enzymes can be constructed to possess the maximum possible k_{cat}/K_M in this scenario, which is 75.0. This value has three components: one for each of the three dimensions of the active site and transition state. In the model as it stands these are completely independent of each other.

For any set of transition states with identical or sufficiently dissimilar axes dimensions one can often use a simple algorithm to specify a proto-enzyme with the highest possible set of activities that encompasses them all. In effect, one creates a single notional transition state to describe all the members of the set. Each of the dimensions of this transition state is assigned the largest value for that dimension from the set. The desired proto-enzyme is the maximum activity enzyme for this notional transition state (i.e. with an active site displaying perfect complementarity towards it).

Admittedly, this procedure will not always generate the best enzyme for the set as a whole (this is the meaning of the 'sufficiently dissimilar' proviso).

It may fail to do so if there a number of substrates with some dimension(s) slightly smaller than the largest. Recall from the shape of the Lennard-Jones function (figure 3.1) that repulsive forces between active site and transition state are felt if the separation between them is less than 89 per cent of R_{max}. If the active site dimensions can be kept sufficiently large to avoid proximity closer than this, an enzyme constructed using the smaller values will not exclude the larger one, and could be a better catalyst over all members of the set. Ignoring this complication gives a simple method for specifying a maximum activity catalyst for any given set of substrates (as long as the requirement does not demand that all substrates not specified in the set must be discriminated against).

The question of discrimination can be illustrated if we return to the real world for a moment. One cannot build an enzyme with a single active site which takes isoleucine as a substrate without also allowing valine as a substrate. It is true that one can design an enzyme to be considerably more active towards isoleucine than valine (because isoleucine has an extra binding group) but to prevent reaction with valine is not possible if the only criterion for discrimination involves the relative interaction energies of the molecules with the enzyme active site.

The activities of the 'perfect' enzymes for the nineteen classes we have encountered are presented in table 6.4.6, where they are compared with the activities evolved in the various experiments. The 'perfect' sequences are shown in figure 6.4.5. The quotes around 'perfect' seem justified, as the evolved enzymes have higher activities more often than not, because several of

the sets of substrates do have distinct dimensions which are sufficiently similar to fail to meet the proviso made above about the effectiveness of the procedure. Having got this far, it would be nice to be able to answer the question: what is the best strategy for these cells? What is the highest realisable growth rate? At the present, it seems that the best advice to the aspiring cell is to say that large genome sizes are better than small ones, and possessing a larger number of classes allows greater control over the relative net activities. The problem with multifunctionality is clear: activities are coupled, and responding to selection for higher growth requires the various levels of activity to be independently manipulable.

Table 6.4.6

	'Perfect' Enzyme								*Best Evolved Enzyme*							
Class	*k_{cat}/K_M (per step)*								*k_{cat}/K_M (per step)*							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
I	0,00	0,00	29,01	40,08	0,00	0,00	0,00	37,76	0,00	0,00	29,25	39,89	0,00	0,00	0,00	38,87
II	0,00	0,00	0,00	3,79	50,41	0,00	32,27	6,22	0,00	0,00	0,00	4,01	50,33	0,00	31,65	6,44
III	0,00	0,00	0,00	3,30	31,17	38,52	26,70	5,31	0,00	0,00	0,00	3,52	31,26	39,75	26,19	5,55
IV	0,00	0,00	25,50	3,29	29,26	0,00	32,16	3,05	0,00	0,00	25,31	3,22	29,15	0,00	32,50	3,10
V	42,53	42,53	0,00	0,00	0,00	29,04	0,00	0,00	45,30	45,51	0,00	0,00	0,00	26,72	0,00	0,00
VI	25,41	0,00	0,00	3,13	26,34	17,36	25,50	5,04	25,04	0,00	0,00	3,44	26,42	19,51	24,33	5,21
VII	0,00	26,42	26,42	15,16	0,00	0,00	0,00	25,72	0,00	26,64	26,63	17,85	0,00	0,00	0,00	23,42
VIII	27,64	0,00	27,64	25,25	8,00	0,00	0,00	12,87	27,82	0,00	27,89	24,50	8,29	0,00	0,00	14,22
IX	25,54	0,00	0,00	6,30	0,00	17,54	26,01	26,20	25,23	0,00	0,00	6,60	0,00	18,81	25,71	26,18
X	0,00	25,12	0,00	0,99	30,79	25,28	2,03	4,27	0,00	25,12	0,00	1,03	31,02	24,79	2,51	4,54
XI	0,00	0,00	27,76	25,90	32,07	0,00	0,00	14,05	0,00	0,00	27,89	25,58	32,18	0,00	0,00	14,96
XII	0,00	0,00	25,41	2,79	10,02	38,46	26,58	2,14	0,00	0,00	25,28	3,28	11,60	34,89	23,96	2,37
XIII	27,80	0,00	0,00	28,92	0,00	0,00	0,00	37,20	27,92	0,00	0,00	28,36	0,00	0,00	0,00	38,19
XIV	47,99	0,00	47,99	0,00	29,77	0,00	0,00	0,00	47,69	0,00	48,23	0,00	29,57	0,00	0,00	0,00
XV	0,00	29,31	0,00	0,00	0,00	32,60	26,85	0,00	0,00	28,94	0,00	0,00	0,00	32,62	26,91	0,00
XVI	0,00	0,00	0,00	43,75	0,00	0,00	0,00	62,09	0,00	0,00	0,00	45,02	0,00	0,00	0,00	61,73
XVII	0,00	50,02	50,02	0,00	0,00	0,00	0,00	0,00	0,00	49,27	48,67	0,00	0,00	0,00	0,00	0,00
XVIII	29,60	0,00	0,00	0,00	0,00	24,69	50,33	0,00	28,18	0,00	0,00	0,00	0,00	27,37	49,03	0,00
XIX	26,08	0,00	0,00	27,46	0,00	18,40	29,18	0,00	26,05	0,00	0,00	27,01	0,00	20,21	27,50	0,00

Pathway Favouring Multifunctionality

Figure **6.4.5**

a) The metabolic pathway of experiments 1, 2, 3, A1, A2, and Introductory Repeated here to ease comparison of transition state and active site dimensions.

b–e) The nineteen classes of proto-enzymes of the final populations of the above-mentioned experiments.

(Dimensions are of the calculated 'perfect' enzymes).

(a).

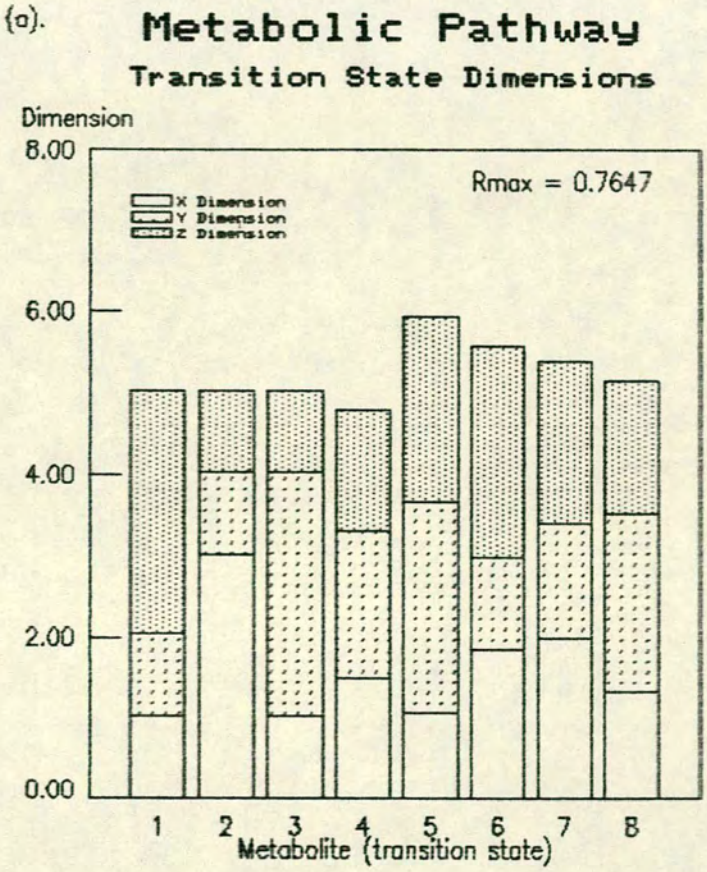
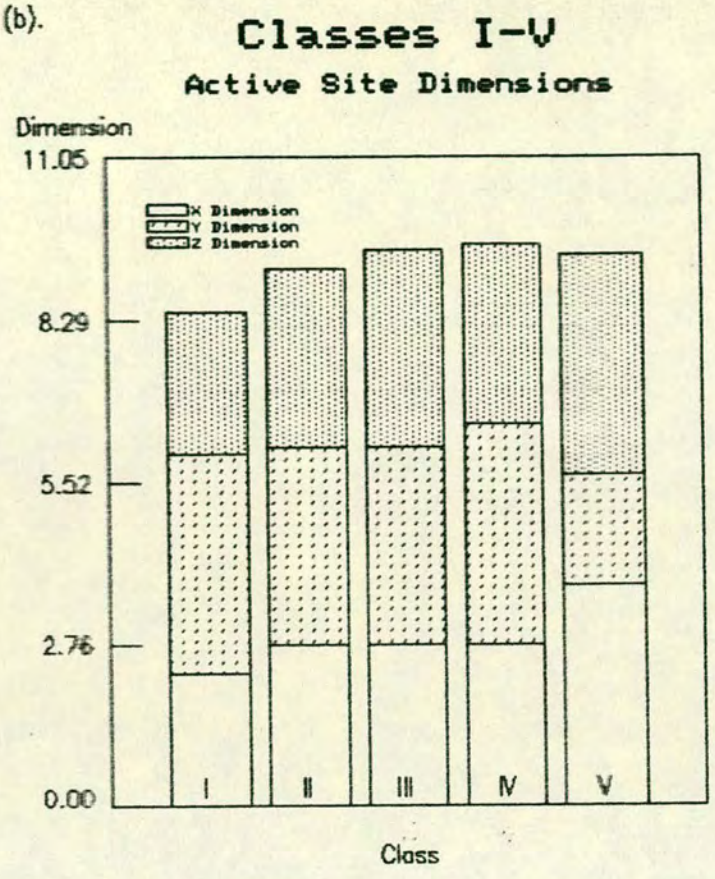


Figure 6.4.5

(b).



(c).

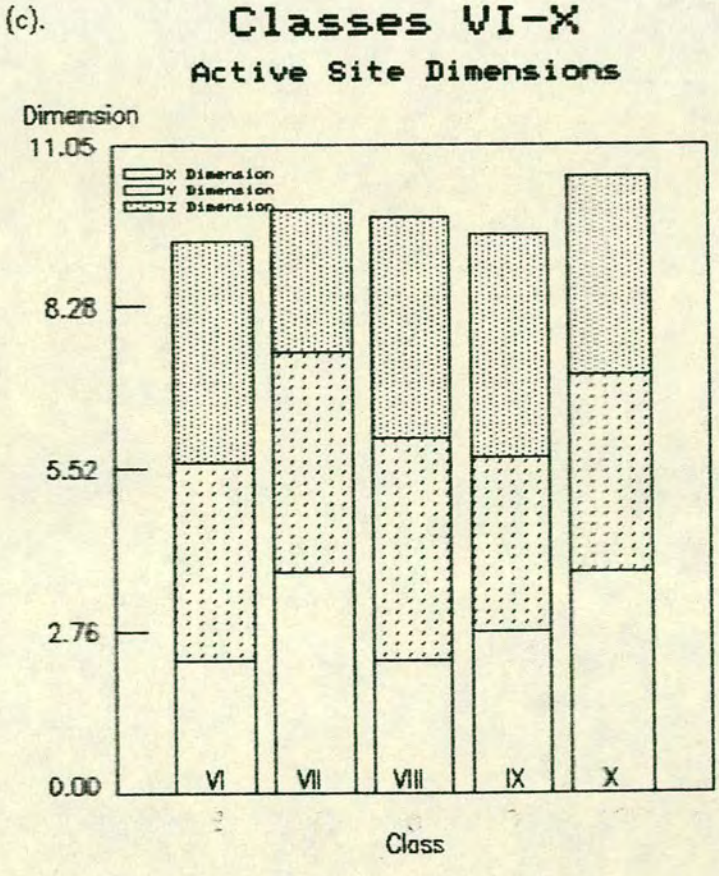
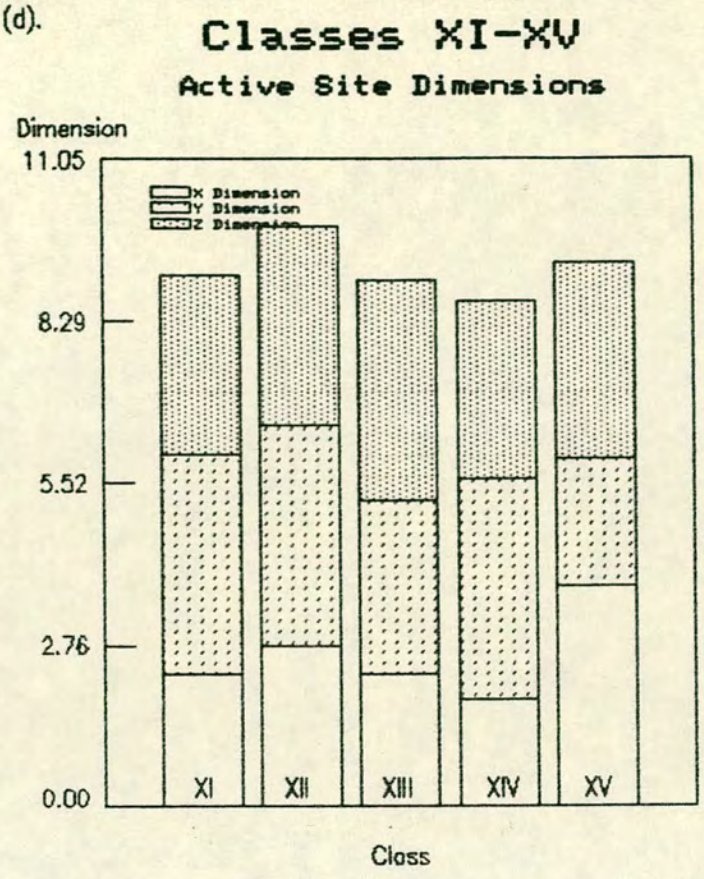


Figure 6.4.5

(d).



(e).

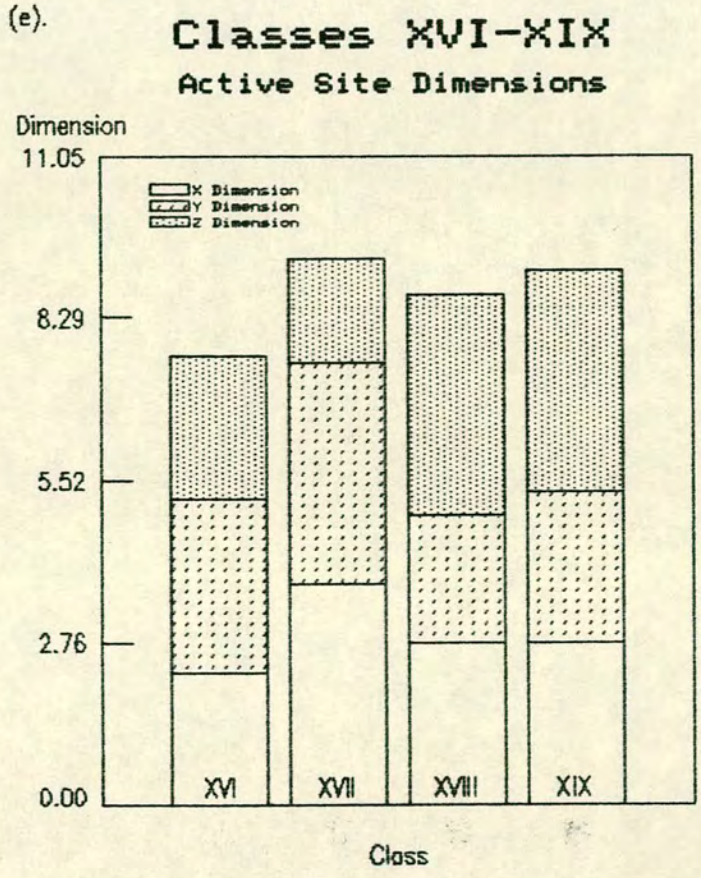


Figure 6.4.5

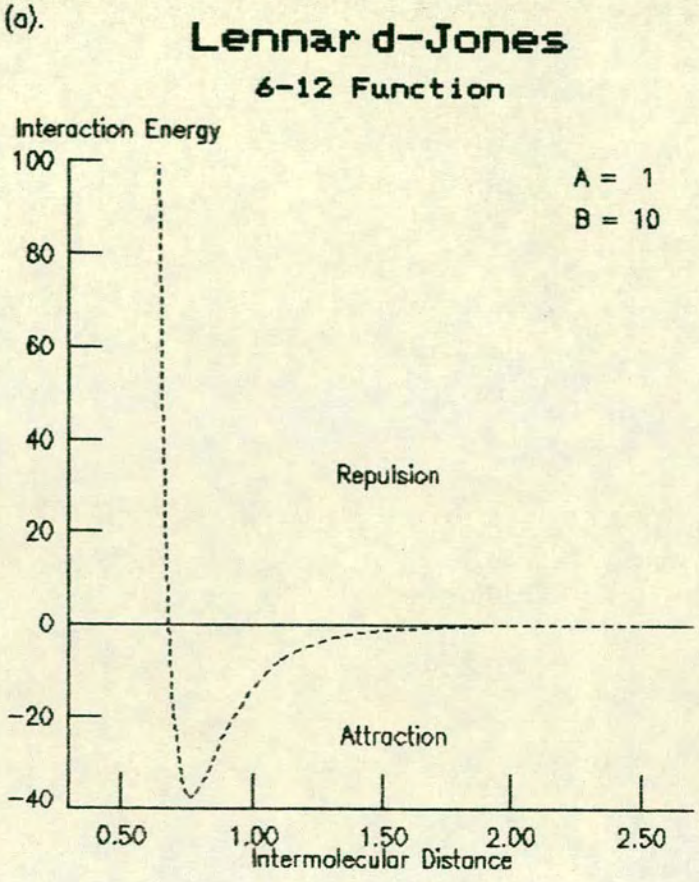
The Effect of the Lennard-Jones Constants

The experiments discussed so far have involved a scenario using the Lennard-Jones function with the values of the constants chosen so that R_{max}, the intermolecular distance yielding maximum binding energy (to which k_{cat}/K_M is directly proportional in the present model) is fairly large compared with the differences in sizes of the different transition states. Figure 6.4.6a shows the Lennard-Jones function for A = 1.0 and B = 10.0 (the values used up to this point). For the present pathway, the mean difference in size between progressively larger transition state dimensions is 0.33 for x and z and 0.29 for y. The shape of the function is such that an enzyme which evolves the x dimension of its active site for maximum interaction with a particular transition state will on average display appreciable activity towards the transition state with the 'next smallest' x dimension (if exclusion due to one of the other dimensions does not occur). This is fairly clear from the figure. If the enzyme active site is 0.76 distance units (this is the turning point of the function on the graph) larger in the x dimension (to pick the worst case) than that of the transition state in question, so that binding energy is maximum between them, then it is 1.09 units larger, on the average, than the x dimension of the transition state with the next largest dimension in that axis. Inspection of the graph shows that a separation of 1.09 yields a significant interaction energy. This accounts for the almost universal catalysis of reactions 4 and 8 in experiments 1, 2, and 3.

What is the effect of retaining the present set of transition states but modifying the value of the constants in the Lennard-Jones formula? If values

are chosen which require much narrower ranges of separation for significant binding energy, then a large part of the advantages of multifunctional proto-enzymes in catalysis of the present pathway is lost. To explore a scenario in which monofunctional proto-enzymes would be a viable option for the cell, the present pathway was retained, but the value of the Lennard-Jones constants were changed to A = 1.0 and B = 10000.0. The results were scaled to give the same maximum binding energy (and hence k_{cat}/K_M) as the previous experiments, and figure 6.4.6b shows the new function. Note that the x-axes of the two figures have the same scale, but different ranges.

(a).



(b).

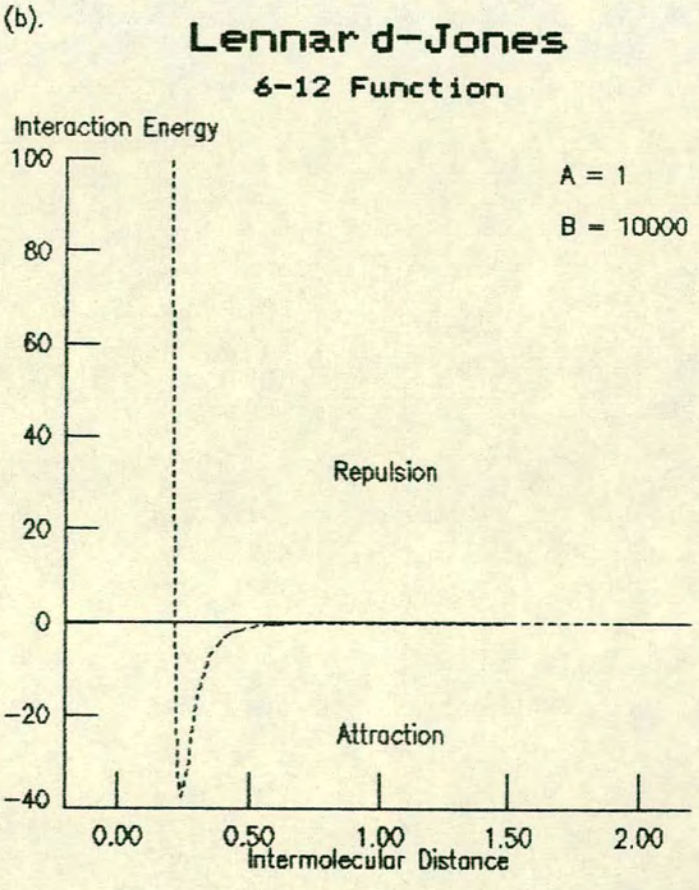


Figure 6.4.6

Pathway with High Lennard-Jones *B* Constant - Experiment 4

The population of the first experiment we shall consider suffered the same fate as that of experiment 2, it became extinct (figure 6.5.1a). The truncation factor throughout the experiment was 1.00. Only a single execution of the simulation program was involved, as the population survived only some 396,000 cell divisions. The mean growth rate at the end of the experiment was 57.707 (figure 6.5.1b), an interesting value as it is a little less than the growth rate of a cell with eight monofunctional proto-enzymes (58.594). The brevity of the simulation (at least by the standards of progress in the previous experiments) means that nothing can really be deduced from the growth rate in isolation.

One of the recurring themes of the previous experiments was the domination of the various populations by cells with identical genome sizes, all deviations from which yield lower growth rates. Like the monofunctional systems with no overlap in activity examined in chapter two, there are optimal allocations of protein to the different proto-enzyme species represented in the cell, and variations in genome size move the cell out of the approximation to that optimum which it has found. Mutations can continue to occur, and these may eventually convert all duplicate-copy genes into unique-copy ones, but such mutations do not alter the functional nature (class) of the proto-enzymes. Only 'micromutations', with very small or zero selective effects, are allowed.

The present population shows this same pattern. Mean genome sizes, after some fluctuation, settle on exactly six (figure 6.5.1c), and all variation in genome

Experiment 4;

Duration: 396,000 cell divisions.

(Terminated by extinction).

Sampled: Every 3000 cell divisions.

Truncation factor: 1.00

Figure 6.5.1

a) Population: one bottleneck
b) Mean growth rate of population
c) Mean genome size & complexity
d) Variation in growth rate
e) Variation in size & complexity

(a).

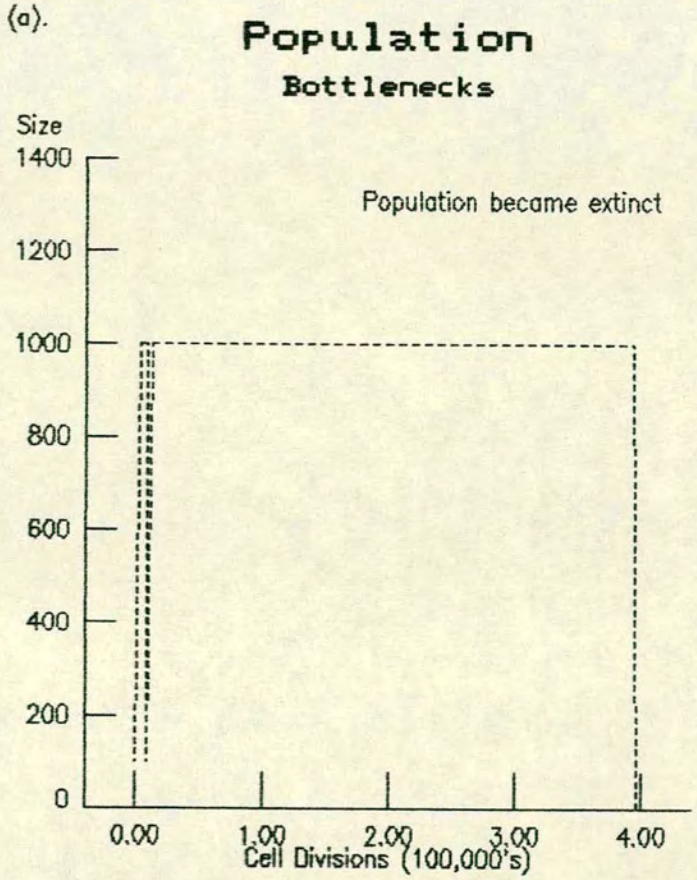
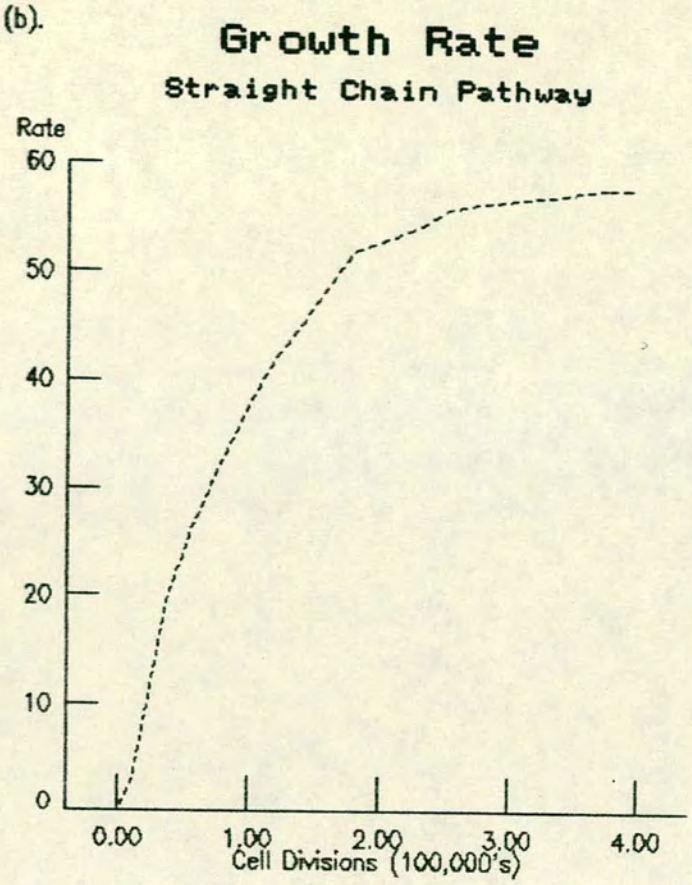


Figure 6.5.1

(b).



(c).

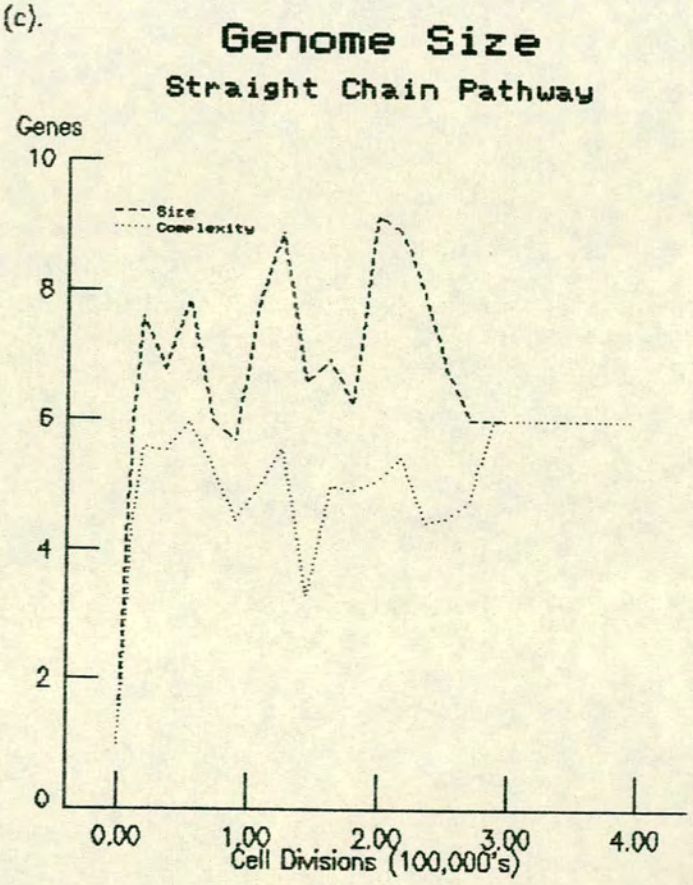
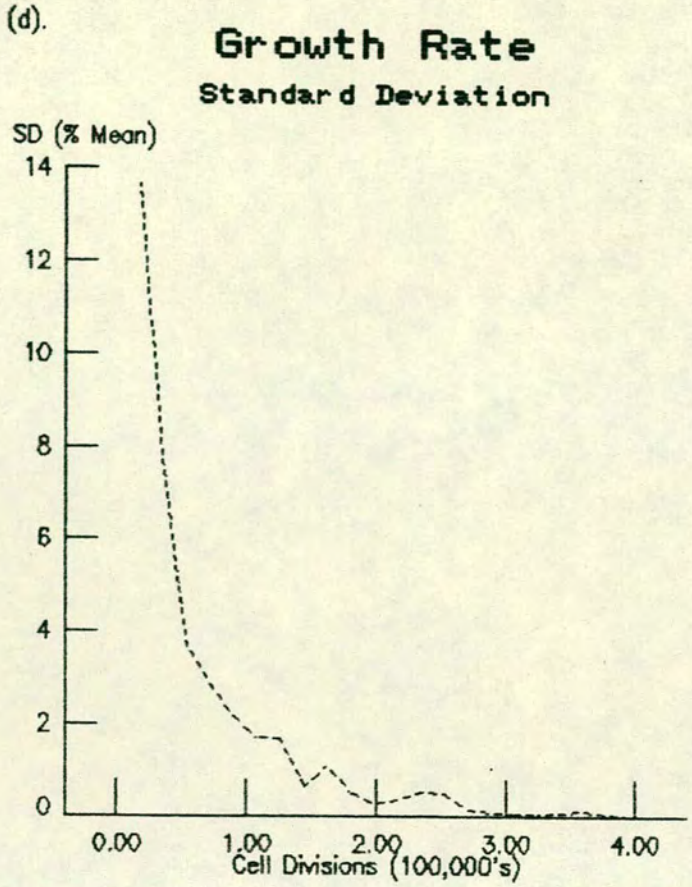


Figure 6.5.1

(d).



(e).

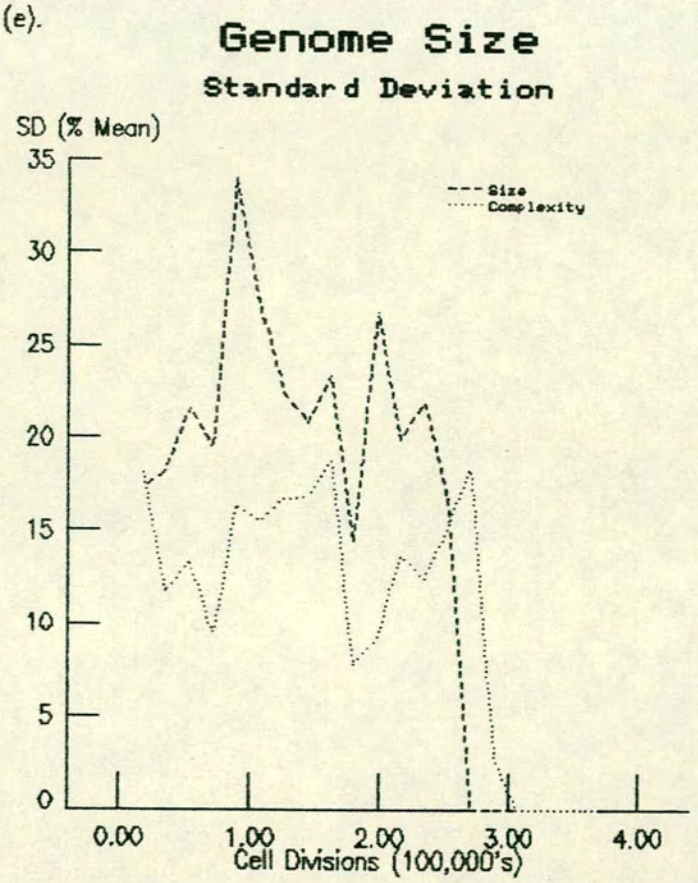


Figure 6.5.1

size is lost (figure 6.5.1e). Some thirty thousand cell divisions later, the remaining variation in the genome complexity is lost, and it too settles at six. (Complete monofunctionality, incidently, has clearly not evolved, as there are not enough proto-enzymes).

Figure 6.5.2a displays the now familiar set of transition states that define the pathway for this experiment, and gives the new value for R_{max}. Maximum binding is now obtained, as expected from figure 6.4.6, at a separation of 0.2418, considerably less than the previous value of 0.7647. In the light of this, it may at first sight seem disconcerting to discover (figure 6.5.2b) that the mean number of activities per catalyst is actually higher in the present experiment than in any previous one! The net extractable activities per unit cell weight (figure 6.5.2c), however, are fairly ordinary, the relatively low values presumably simply reflecting the relatively brief evolutionary history of the cells.

Examination of the individual activity profiles of the cells (figure 6.5.3) is surprising. After 100,000 cell divisions, the mean number of activities per proto-enzyme is something over five. However, the cell shown in figure 6.5.3a as sampled at that point shows exactly three activities for each of its four proto-enzymes. This discrepancy is not because the cell, being the fastest sampled at this point, is atypical, though similar charts for some other sampled cells would display even less activities (as a few proto-enzymes show only two). The discrepancy arises because many of the reported activities are now so low that, like the activities of the ancestral enzyme of the previous experiments, they are not visible on the chart at all (less than 0.1 arbitrary

units, say) but are included in the statistics which go into figure 6.5.2b. The number of significant activities per protoenzyme has indeed decreased. Nevertheless, the population has progressed a considerable way towards the growth rate of an exclusively monofunctional cell without, for example, developing a single k_{cat}/K_M even as large as 30.0 (figure 6.5.3, b-e), let alone the 75.0 of the 'perfect' monofunctional enzyme. The prospects of the present scenario favouring the evolution of monofunctional enzymes do not, on the basis of this first exposure, seem great.

The reader may, incidently, have noticed that the catalytic activities and growth rate of the ancestral cell of the present experiment (figure 6.5.3a) are somewhat higher than in the previous experiments. This is because the universal catalyst with which it is endowed has a smaller active site than the catalyst in those experiments. The growth rate of the ancestral cell of the previous experiments, with the new Lennard-Jones values, if expressed in the fixed point, five decimal place notation of figure 6.5.3a, would be 0.00000, as the dimensions of the active site for that enzyme are larger than any transition state dimension by at least 4 units (off the x-axis of figure 6.4.6).

The activity profiles shown for the cells sampled at 100,000 and 200,000 cell divisions are quite different but comparing the cells at 280,000 and 370,000 divisions reveals considerable similarity. The earlier cell, sampled shortly before the six/six size and complexity standard was adopted, displays a six/six profile, and it is tempting to theorize about the relationship of the cell to those that later gained ascendance in the population. The six genes

shown in 6.5.3d belong to the same classes as those of the later cell shown in 6.5.3e, with the exception of gene 2. A single mutation (in *z*) would, however, move the gene from its present class into the other. Perhaps this occurred in this cell and we are looking at the ancestor of those which followed, but perhaps this is frivolous and we should heed the warnings of Patterson, 1981, about making hypotheses about ancestral relationships.

As the experiment foundered with a truncation factor of 1.00, the experiment was repeated with slightly weaker selection, and the results of this shall now be discussed.

Experiment 4;

Figure 6.5.2

a) Metabolic pathway – Eight reactions
Same transition states: smaller Rmax

b) Mean number of reactions catalysed by each enzyme.

c) Extractable enzyme activities from single cells sampled when stated.

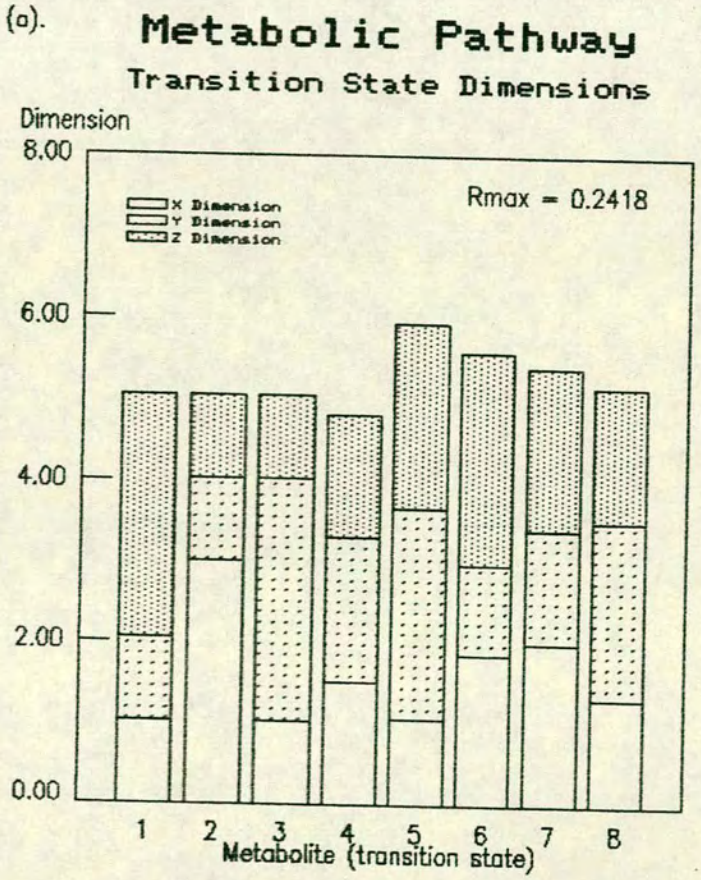
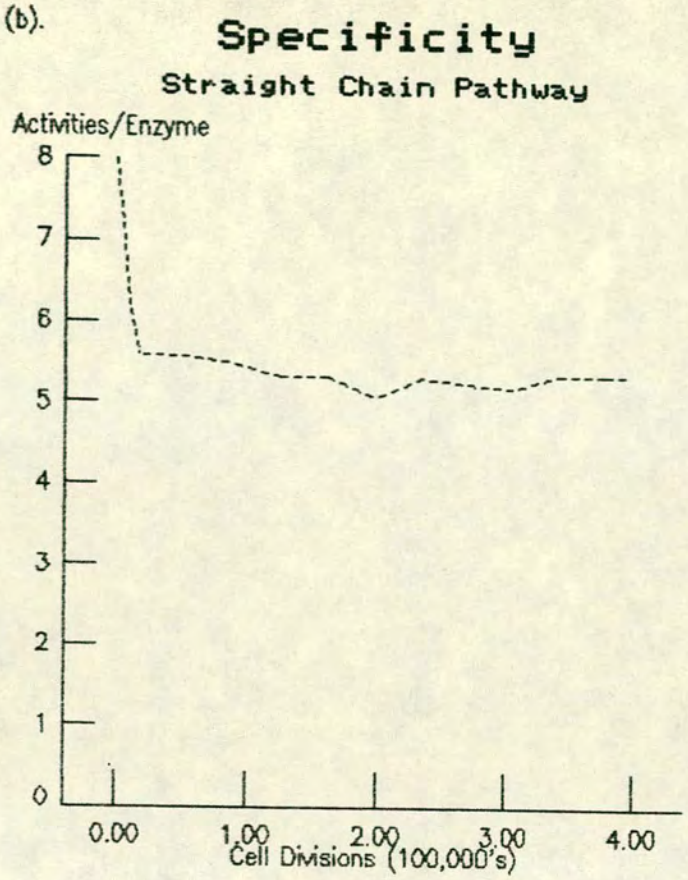


Figure 6.5.2

(b).



(c).

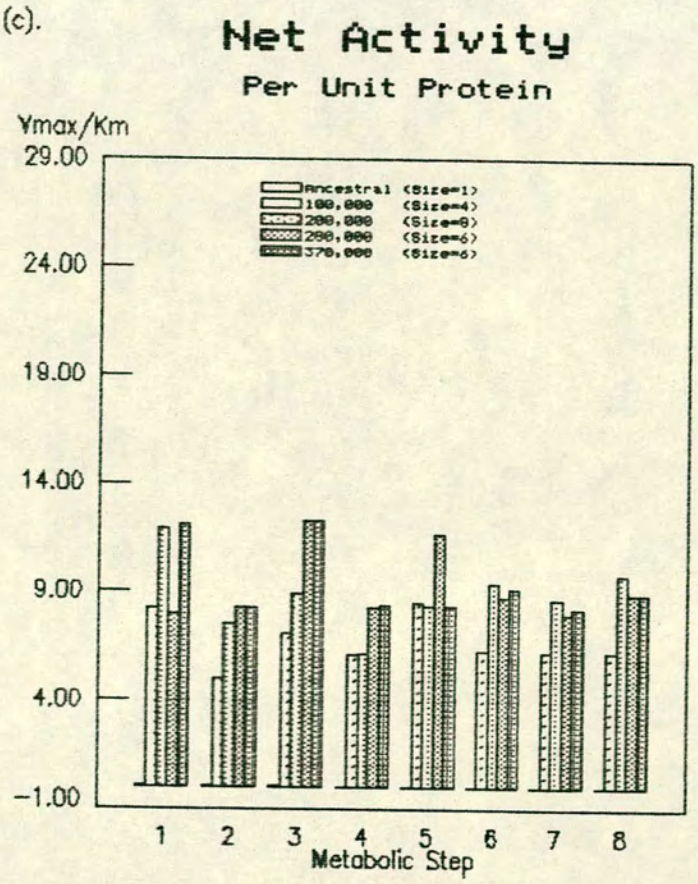


Figure 6.5.2

Experiment 4;

Figure 6.5.3

Profile of the enzyme activities of the fastest cell sampled after:

a) 0 cell divisions (original cell).

b) 100,000 cell divisions.

c) 200,000 cell divisions.

d) 280,000 cell divisions.

e) 370,000 cell divisions.

(a).

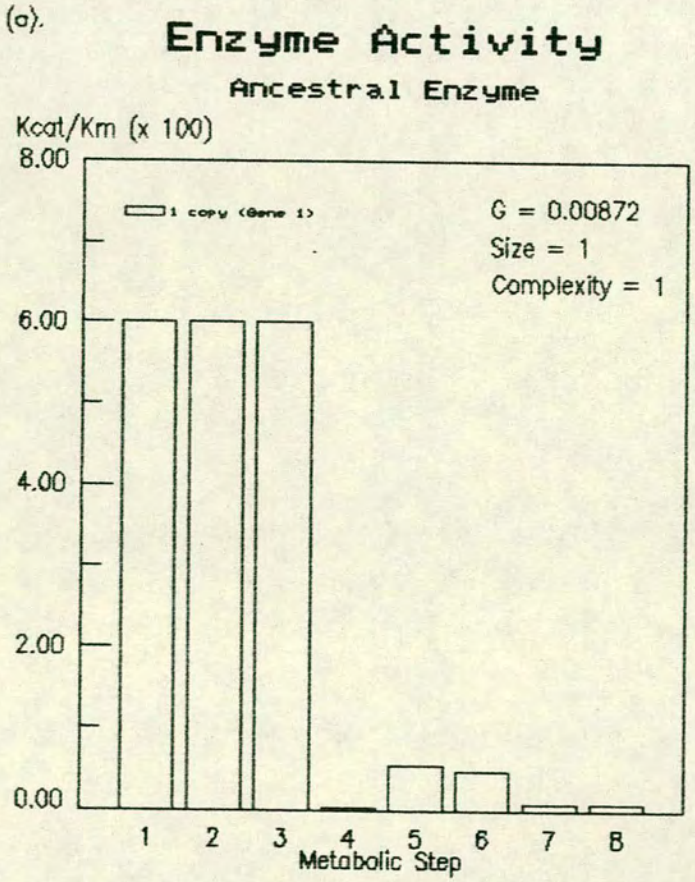
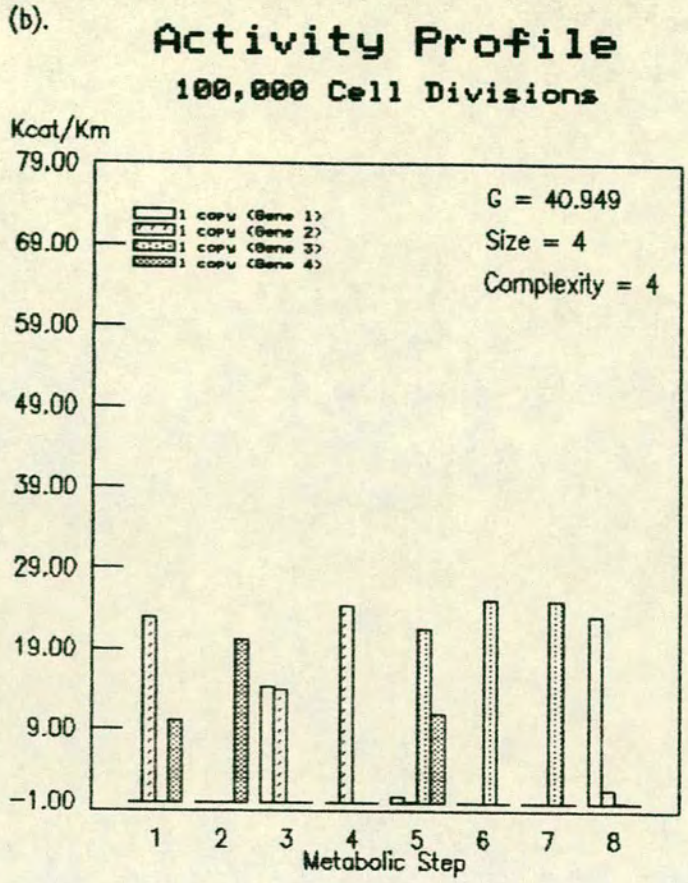


Figure 6.5.3

(b).



(c).

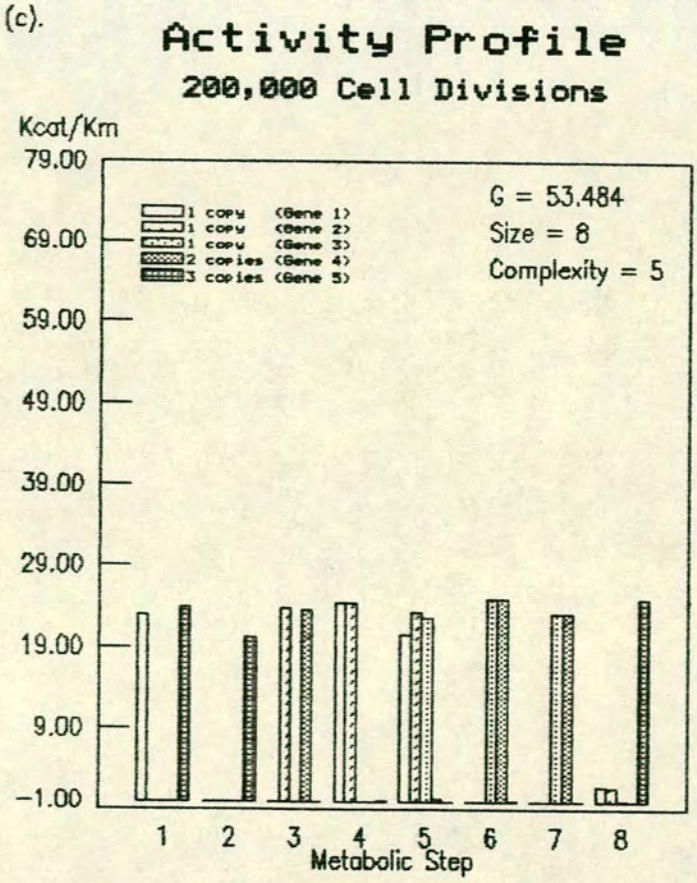
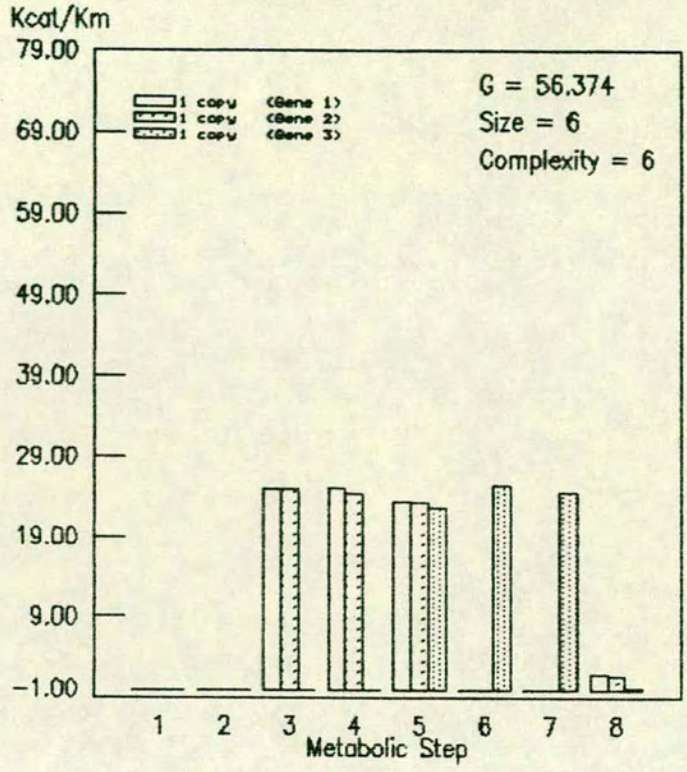


Figure 6.5.3

(d1). **Activity Profile 1-3**

280,000 Cell Divisions



(d2). **Activity Profile 4-6**

280,000 Cell Divisions

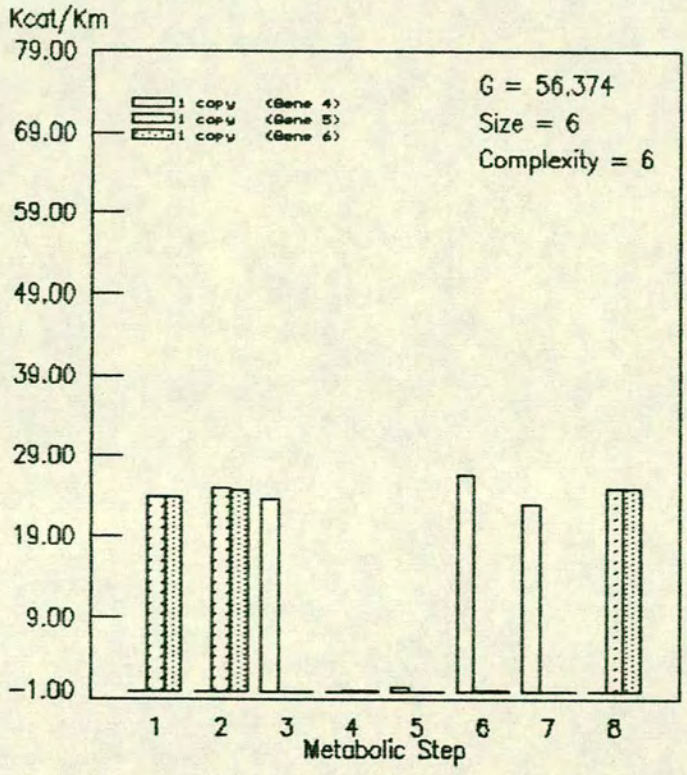
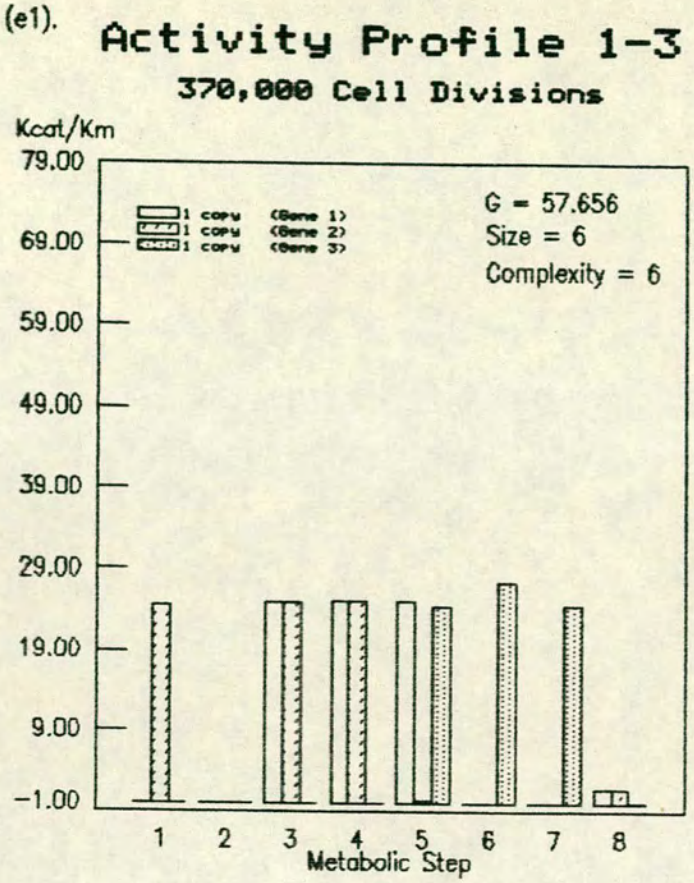


Figure 6.5.3

(e1).



(e2).

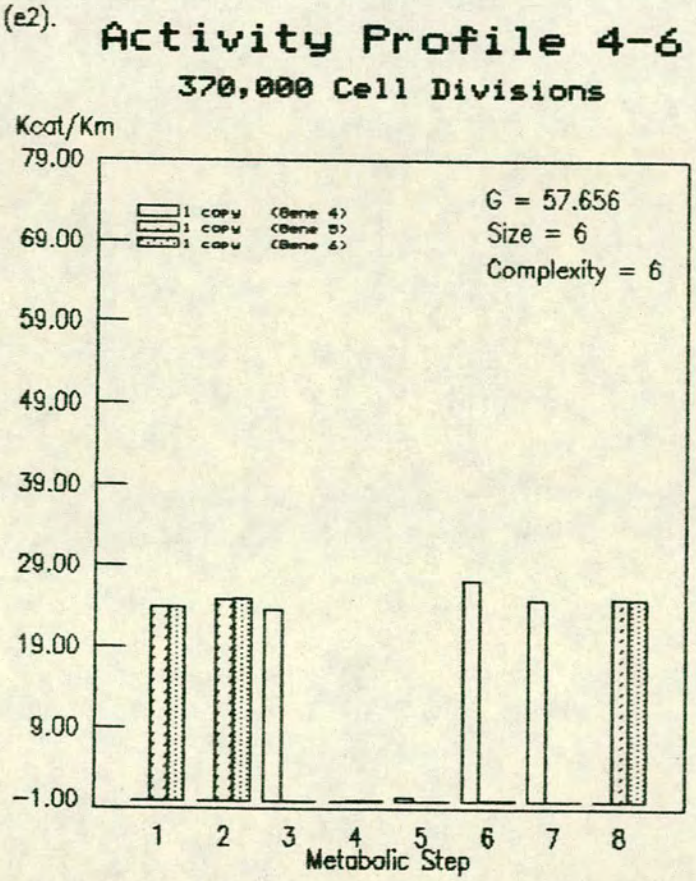


Figure 6.5.3

High Lennard-Jones *B* Constant - Experiment 5

This repeat of experiment 4 is the last straight-chain experiment to be described that involved the set of reaction intermediates first shown in figure 6.1.2a (with the smaller R_{max} shown in figure 6.4.5a). Of all the experiments described so far, this one is most in need of further continuation. The experiment extends for two and a half million cell divisions over four 'sessions' (figure 6.6.1a). Like experiment 1, the experiment began with fairly weak selection initially (truncation factor, 1.01), but at the end of the first run, when growth rate seemed to be levelling off at around 52 (figure 6.6.3b), the intensity of truncation selection was increased by reducing the truncation factor to 1.0005. The effect of this is visible from both the growth rate and the abrupt change in the pattern of cell mortality (figure 6.6.1f). The final growth rate slightly exceeds that of a cell with eight monofunctional proto-enzymes.

A second resemblance with experiment 1 is the rapid rise in mean genome size to high levels. Unlike that experiment, however, where the genome size had peaked (at over 60) and then rapidly fallen to an invariant 23 within the first two and half million cell divisions (figure 6.2.1c), in the present simulation the size was clearly still increasing at the end of the final session (figure 6.6.1c) at which point it was about 80. Considerable variation in both the size and complexity of genomes remained (figure 6.6.1e). The mean genome complexity behaved somewhat differently than in experiment 1, not following the steep upward motion of the mean genome size, but climbing fairly

292 ⇒

Experiment 5;

Duration: 2.46 million cell divisions.

Sampled: Every 3000 cell divisions.

Truncation factor: 1.01 then 1.0005

***Figure* 6.6.1**

a) Population: Bottlenecks of 100

b) Mean growth rate of population

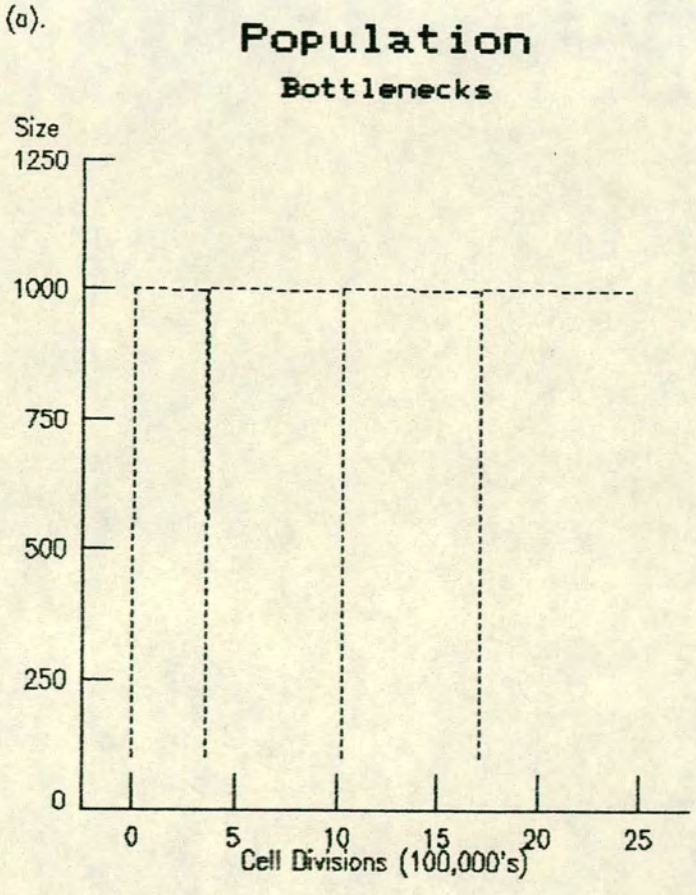
c) Mean genome size & complexity

d) Variation in growth rate

e) Variation in size & complexity

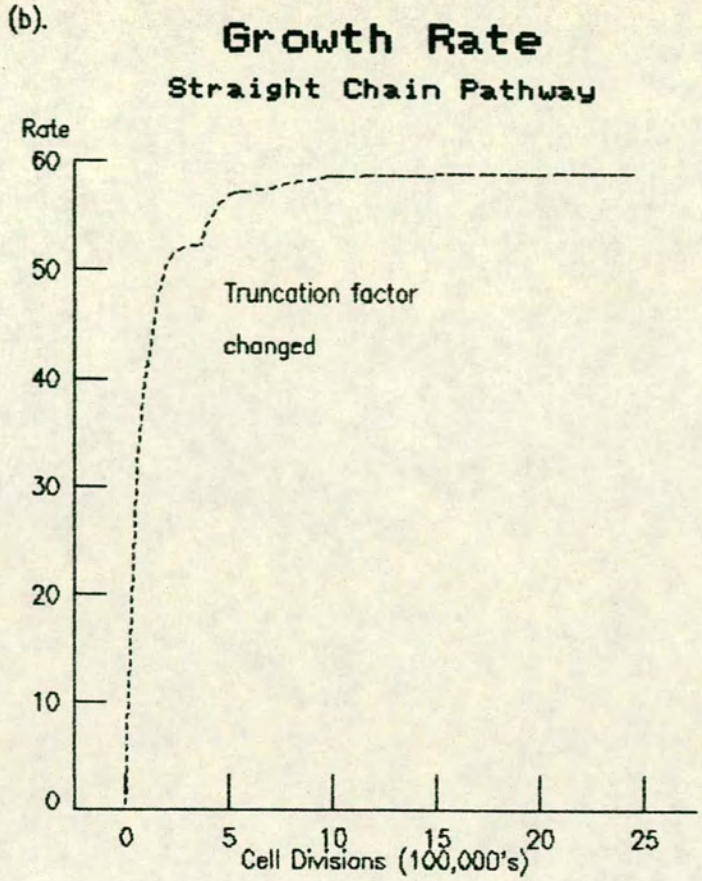
f) Cell deaths from all causes

(a).



igure 6.6.1

(b).



(c).

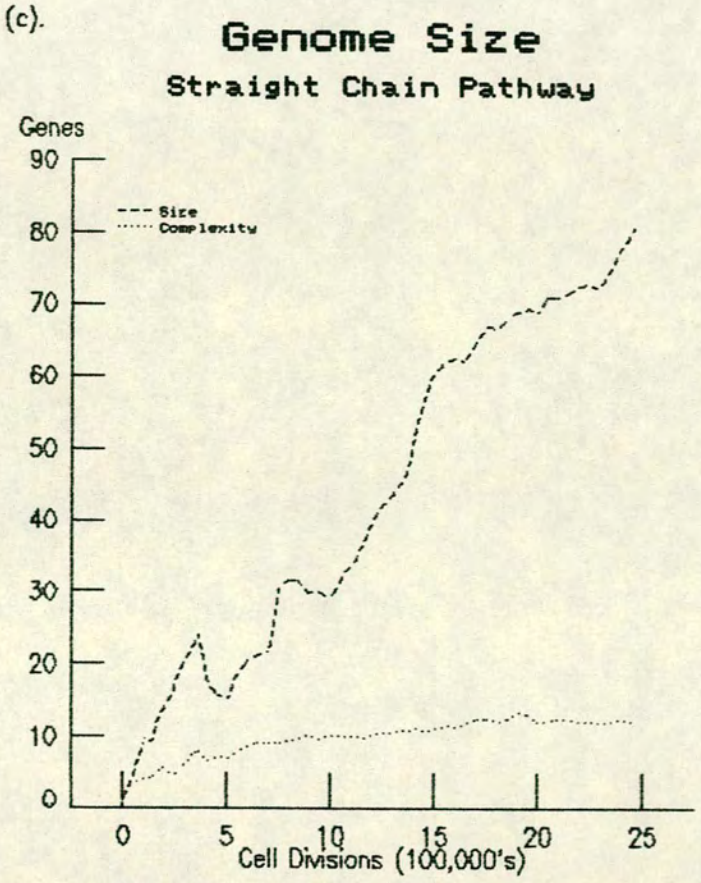
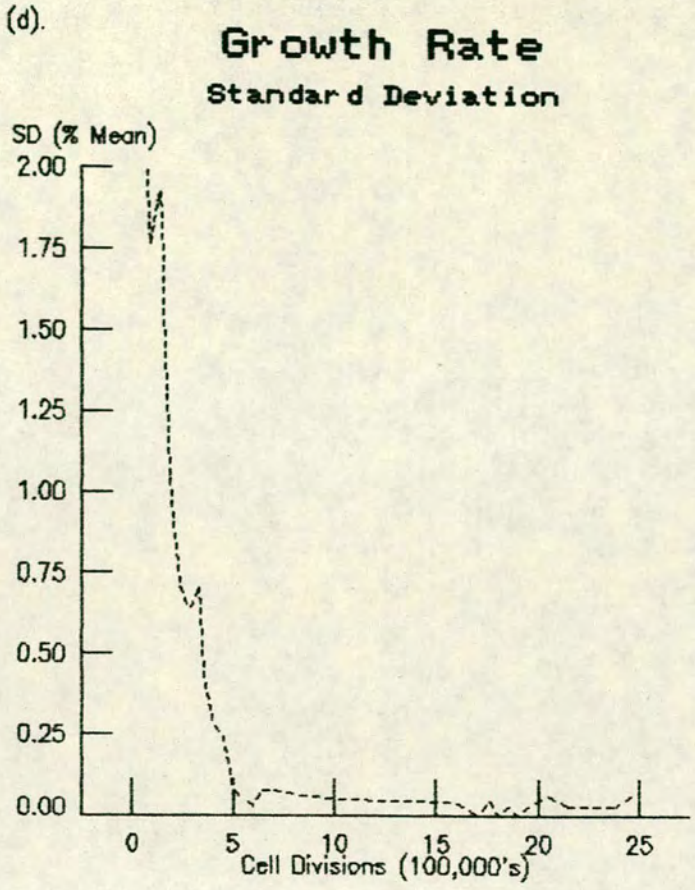


Figure 6.6.1

(d).



(e).

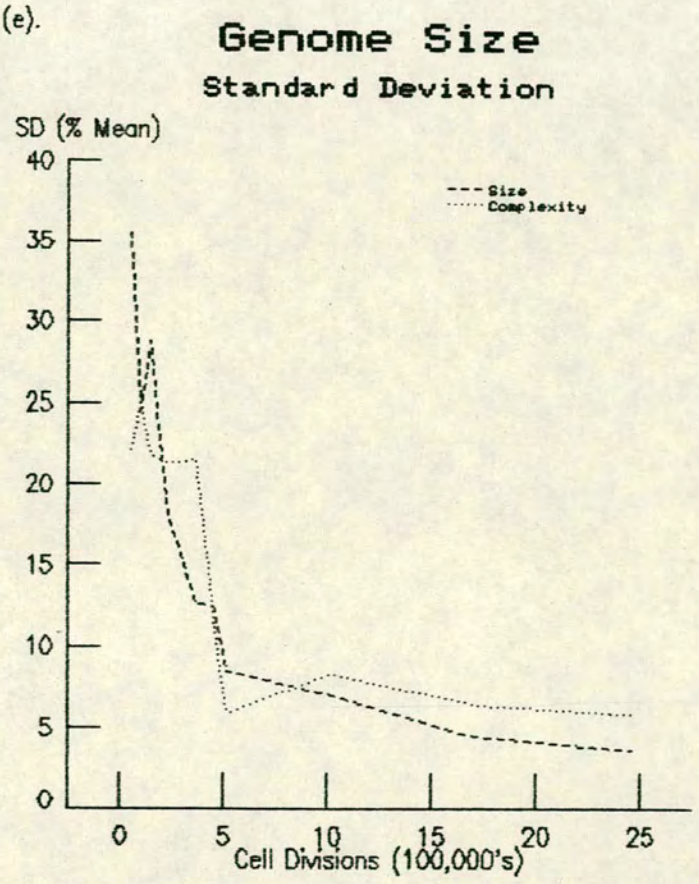


Figure 6.6.1

Experiment 5

(f).

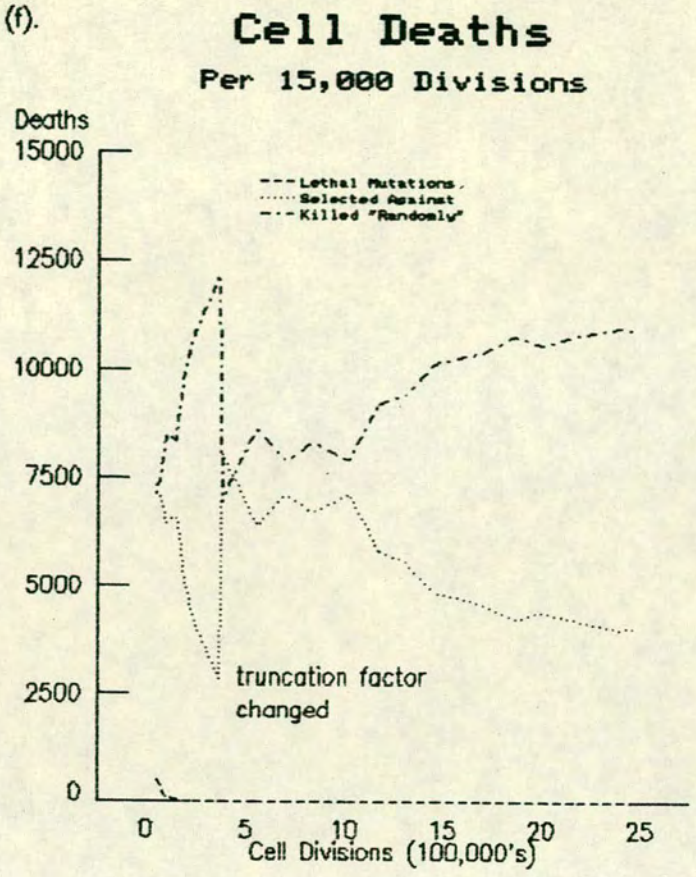

Cell Deaths
Per 15,000 Divisions
Deaths
15000
12500
10000
7500
5000
2500
0
Lethal Mutations
Selected Against
Killed "Randomly"
truncation factor
changed
0
5
10
15
20
25
Cell Divisions (100,000's)

Figure 6.6.1

slowly during the first half of the experiment, and then more or less settling down at about twelve genes.

Variation in growth rate (figure 6.6.1d) settled to a rather lower level than in previous experiments. The truncation factor of 1.0005 was, nevetheless, sufficiently generous that the population was in no apparent danger of extinction, with more than four-fifths of all fatalities being due to overpopulation (figure 6.6.1f). Lethal mutations disappeared very early in the experiment, undoubtedly because of the large genome sizes, which guaranteed that each metabolic step was catalysed by the products of several genes.

The number of activities per proto-enzyme was lower than experiment 4, and much more like the earlier experiments when B = 10 (figure 6.6.2a). After one million cell divisions the levels of extractable activities were comparable to what they would be in a cell with only monofunctional proto-enzymes (figure 6.6.2b). In the earliest sampled cell shown in the figure, there was a very high activity for step 2, but activities 3 to 8 are all low.

Data for cells sampled at the end of each of the four sessions are presented in figure 6.6.3. After 360,000 cell divisions, the genome size was already large, that for the sampled cell being twenty. This cell had eight proto-enzymes. The four with smallest x-axes catalyse reactions 1 and 7, and had additional activity for steps 4 or 8 (significant activity for these seems to be mutually exclusive) (figure 6.6.3a1). Low activity for step 6 was also found in all four of these proto-enzymes. The larger x-axis enzymes were more heterogeneous but have two (in the case of the product of the fifth gene) or

(f).

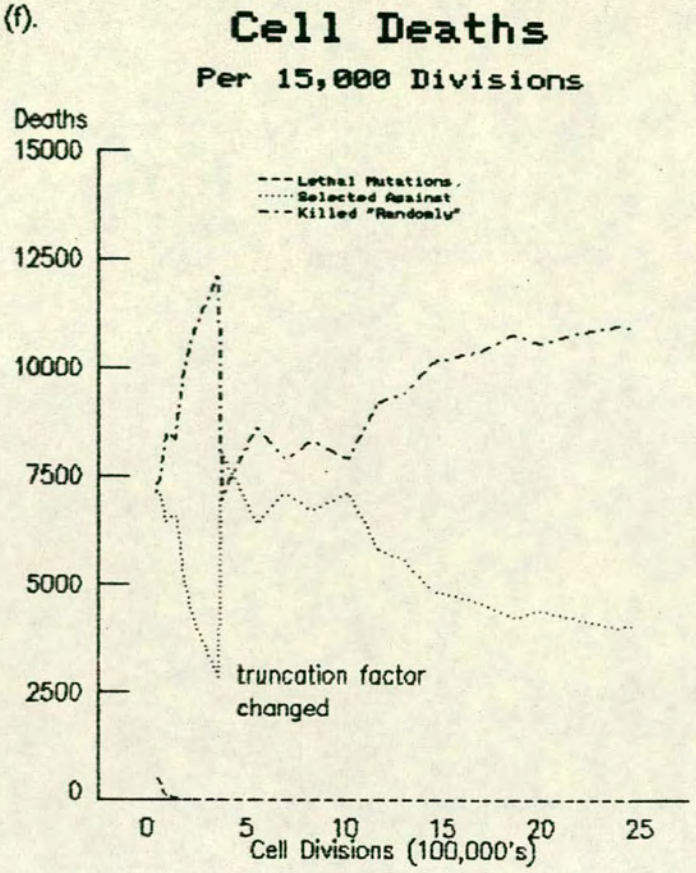

Cell Deaths
Per 15,000 Divisions
Deaths
15000
12500
10000
7500
5000
2500
0
Lethal Mutations
Selected Against
Killed "Randomly"
truncation factor
changed
0
5
10
15
20
25
Cell Divisions (100,000's)

Figure 6.6.1

slowly during the first half of the experiment, and then more or less settling down at about twelve genes.

Variation in growth rate (figure 6.6.1d) settled to a rather lower level than in previous experiments. The truncation factor of 1.0005 was, nevetheless, sufficiently generous that the population was in no apparent danger of extinction, with more than four-fifths of all fatalities being due to overpopulation (figure 6.6.1f). Lethal mutations disappeared very early in the experiment, undoubtedly because of the large genome sizes, which guaranteed that each metabolic step was catalysed by the products of several genes.

The number of activities per proto-enzyme was lower than experiment 4, and much more like the earlier experiments when B = 10 (figure 6.6.2a). After one million cell divisions the levels of extractable activities were comparable to what they would be in a cell with only monofunctional proto-enzymes (figure 6.6.2b). In the earliest sampled cell shown in the figure, there was a very high activity for step 2, but activities 3 to 8 are all low.

Data for cells sampled at the end of each of the four sessions are presented in figure 6.6.3. After 360,000 cell divisions, the genome size was already large, that for the sampled cell being twenty. This cell had eight proto-enzymes. The four with smallest x-axes catalyse reactions 1 and 7, and had additional activity for steps 4 or 8 (significant activity for these seems to be mutually exclusive) (figure 6.6.3a1). Low activity for step 6 was also found in all four of these proto-enzymes. The larger x-axis enzymes were more heterogeneous but have two (in the case of the product of the fifth gene) or

three (genes six to eight) significant activities among steps 2, 3, 4, 5, and 6 (figure 6.6.3a2). The single-copy fifth gene does not seem a good investment, its product having identical activity with the product of the sixth gene (present in five copies) for steps 3 and 5, but completely without 'compensation' for lacking that proto-enzyme's activity for step 2.

After the second session, at roughly one million cell divisions, there were ten proto-enzymes present, two of which (the products of the first two genes) showed nearly identical activity profiles to each other (figure 6.6.3b). No proto-enzyme with activity for only steps 3 and 5 is now present, but proto-enzmes with activities for steps 2, 3, and 5 are represented. The mean genome size had risen to almost thirty (29.8) at this point, with the cell shown having 29 genes in total. At 58.75, the growth rate of the cell is already greater than that of a cell with monofunctional proto-enzymes (58.59).

Seven hundred thousand cell divisions later, the mean genome size had risen to 64.9, and the mean genome complexity to 12.5. The cell shown in figure 6.6.3c at this point had thirteen proto-enzymes coded by sixty-four genes. Two of the thirteen distinct sequences are single-copy. Growth rate has increased by only 0.25 per cent, and the pattern of activity is very similar to that of the cell sampled at one million cell divisions (see below).

At the end of the final session, after two and a half million cell divisions, the mean growth rate had risen by a further 0.30 per cent, to just over 59. The growth rate of the cell shown in figure 6.6.3d is equal to the mean for the

302 ⇒

Experiment 5;

Figure 6.6.2

a) Mean number of activities per proto-enzyme

b) Net extractable activities of cells sampled at the points stated

Extractable activity of a cell with eight monofunctional proto-enzymes is included for comparison.

Figure 6.6.3

Typical cells sampled after

a) 360,000

b) 1,000,000

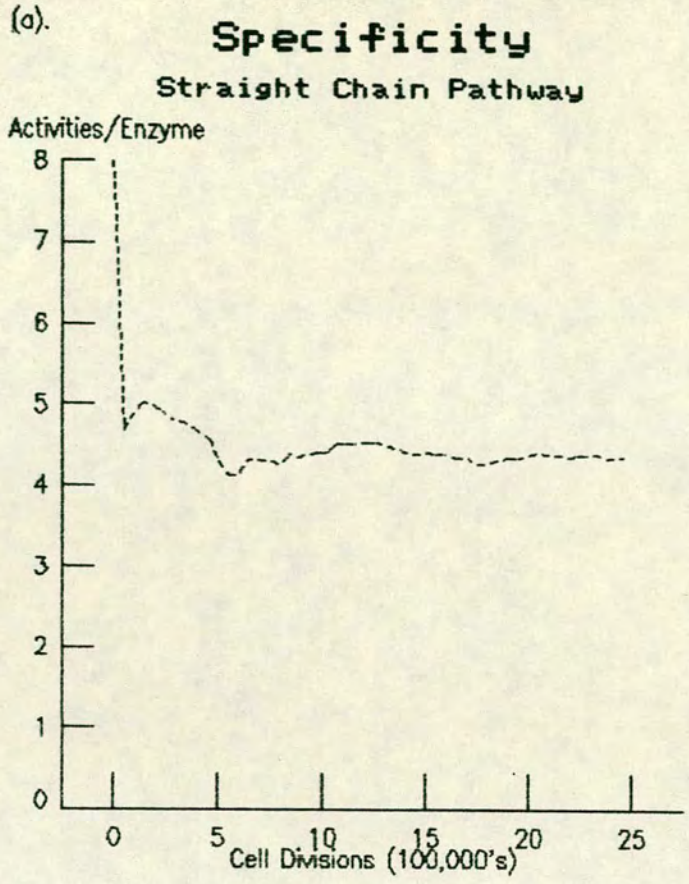
c) 1,700,000

d) 2,460,000

cell divisions.

(These are drawn from the final populations of each of the four runs).

(a).



(b).

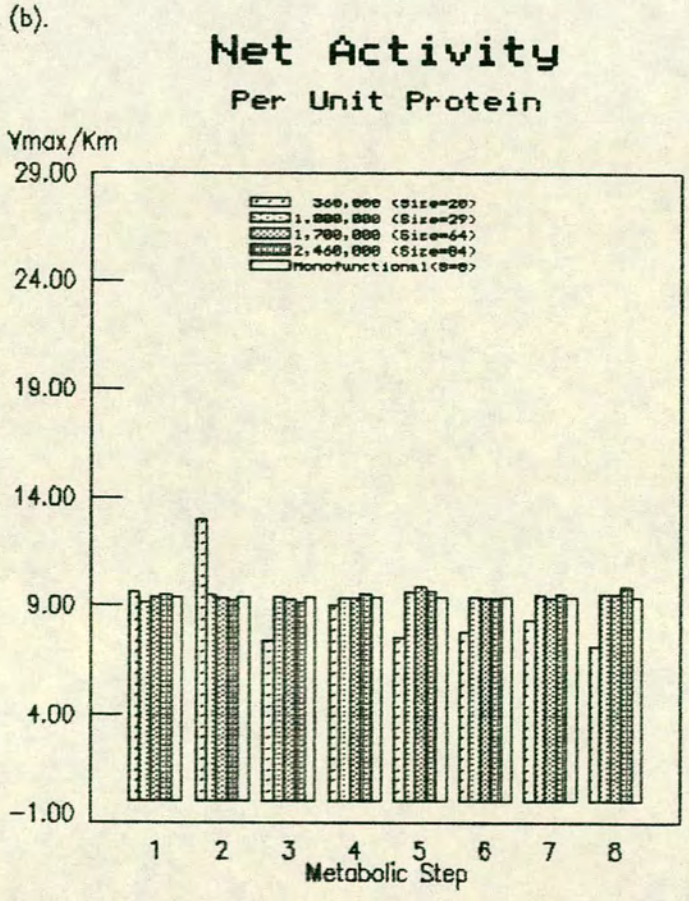
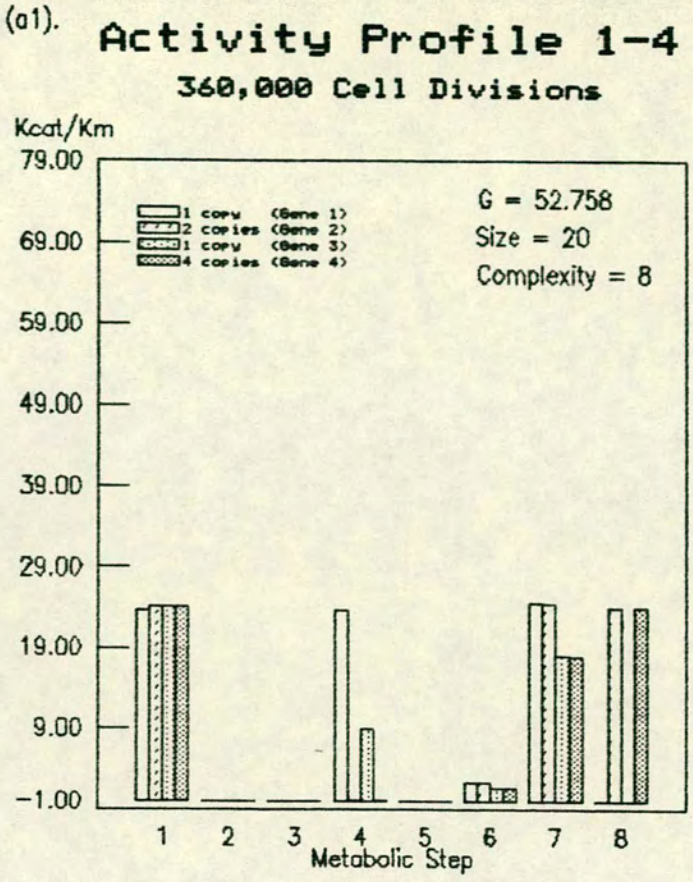


Figure 6.6.2

(a1).



(a2).

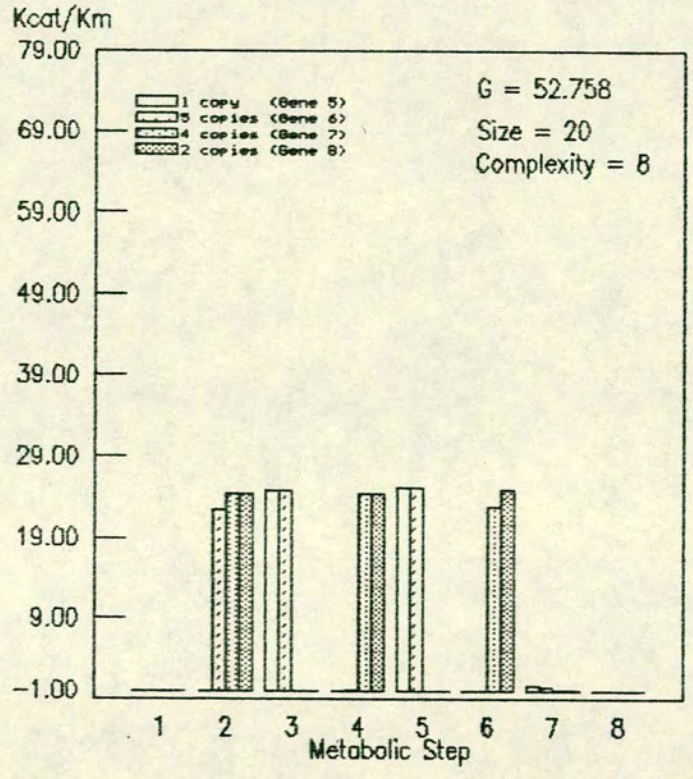
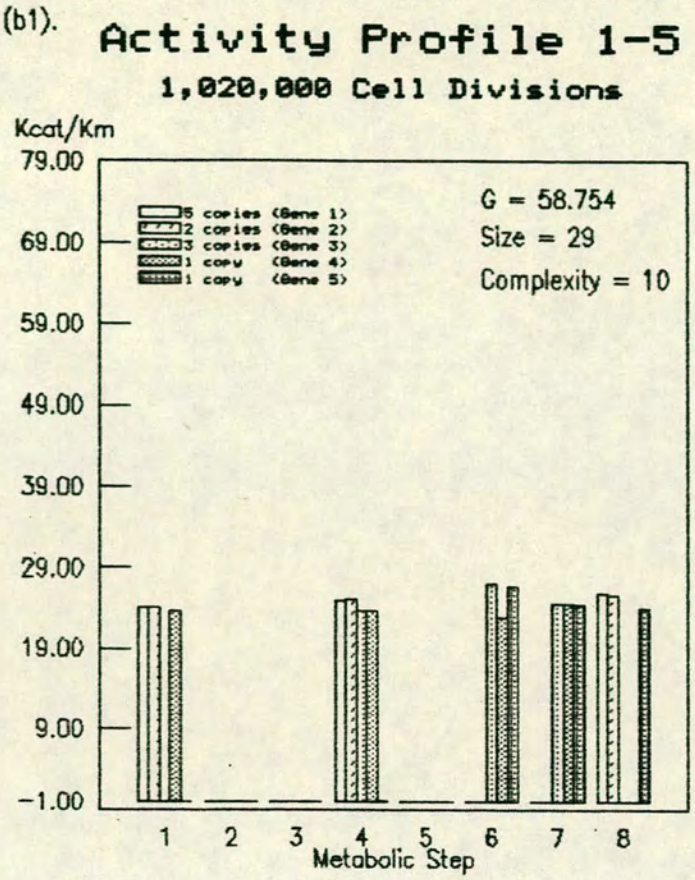


Figure 6.6.3 *Experiment 5* 296

(b1).



(b2).

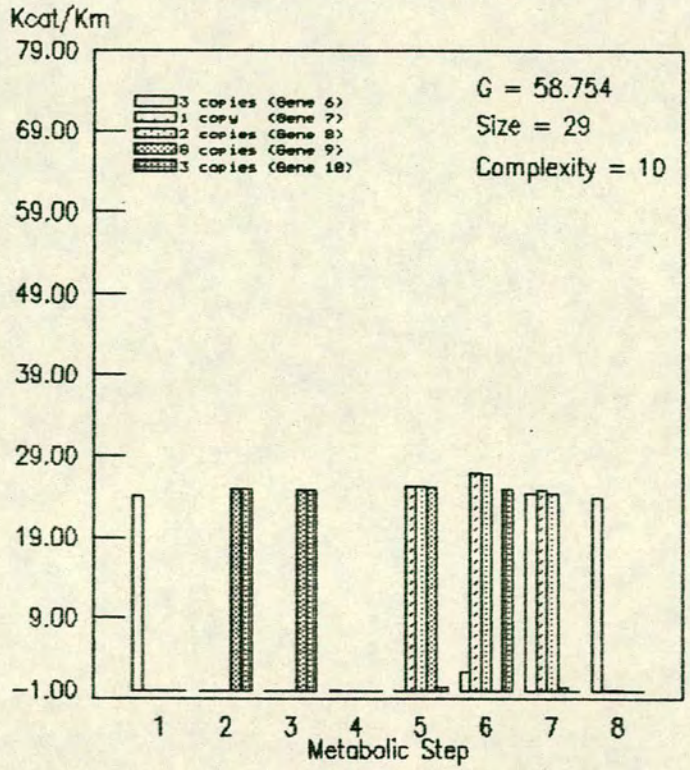
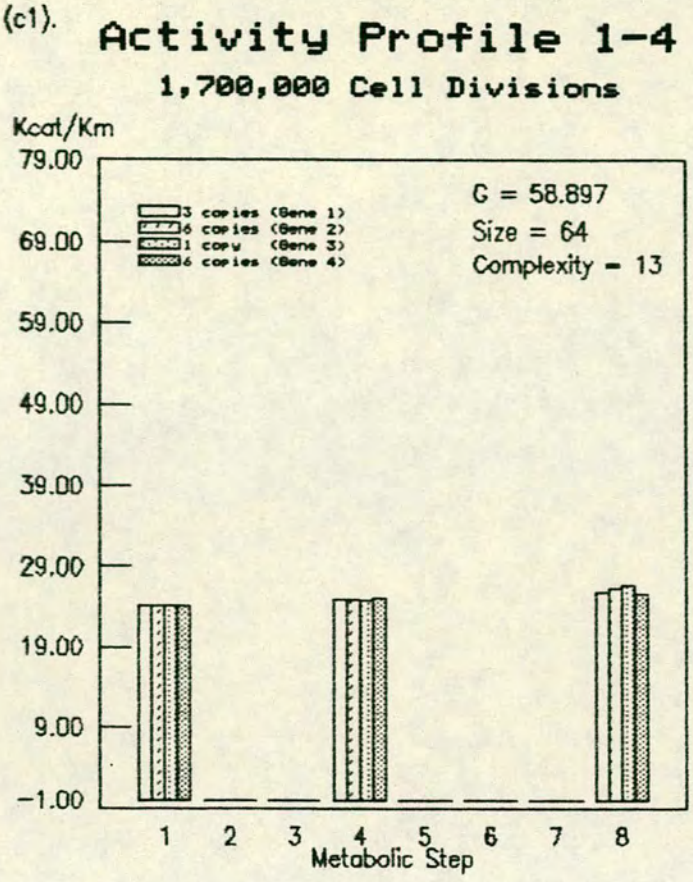


Figure 6.6.3

Experiment 5

(c1).



(c2).

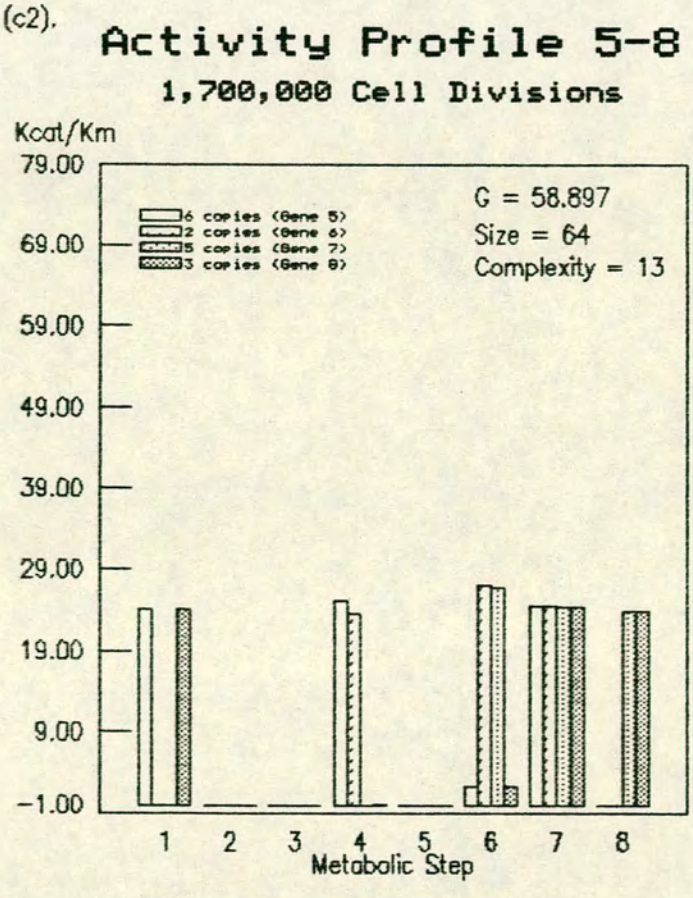


Figure 6.6.3

(c3).

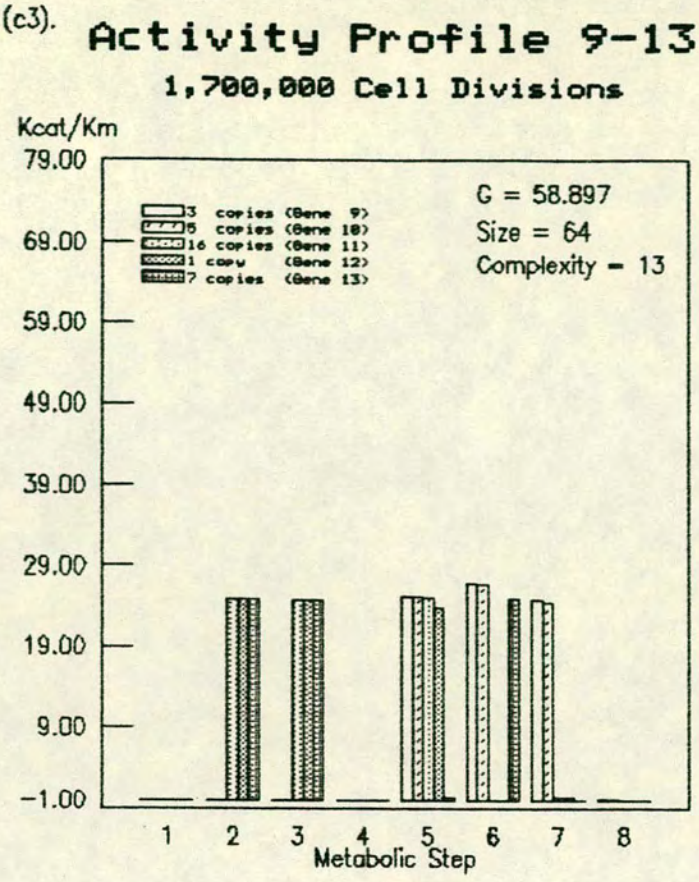
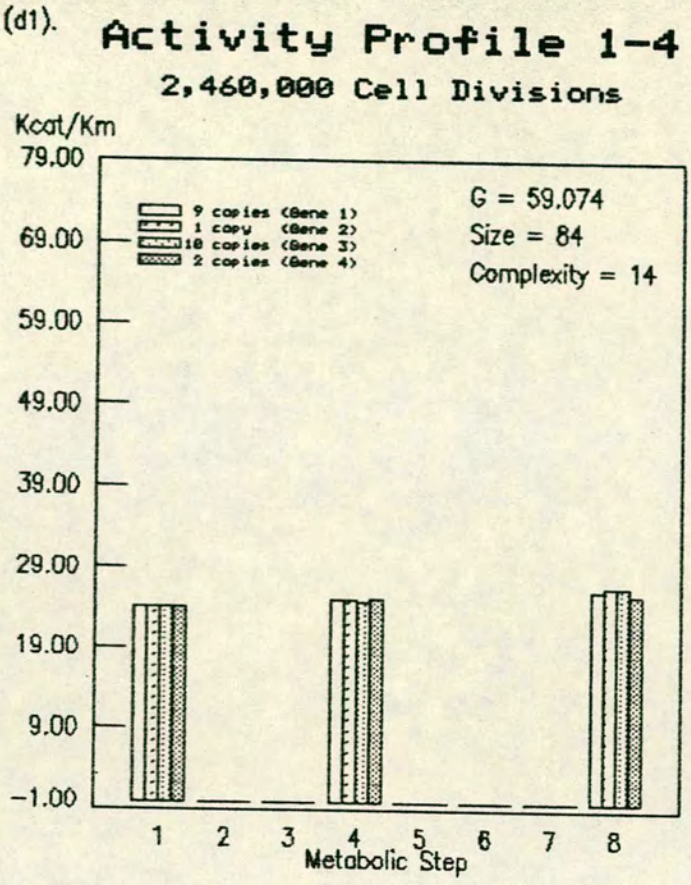


Figure 6.6.3

(d1).



(d2).

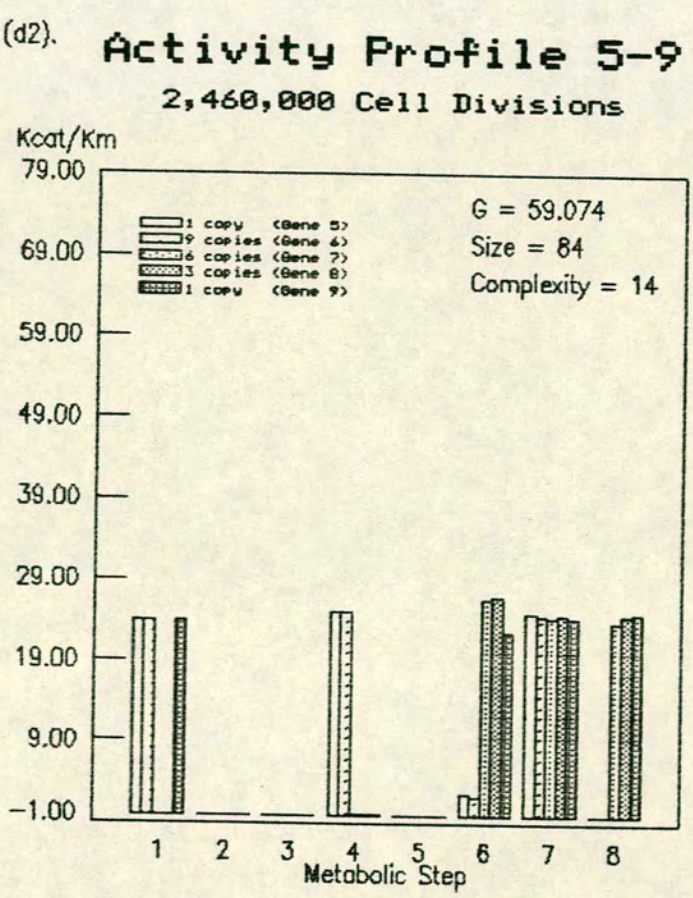


Figure 6.6.3

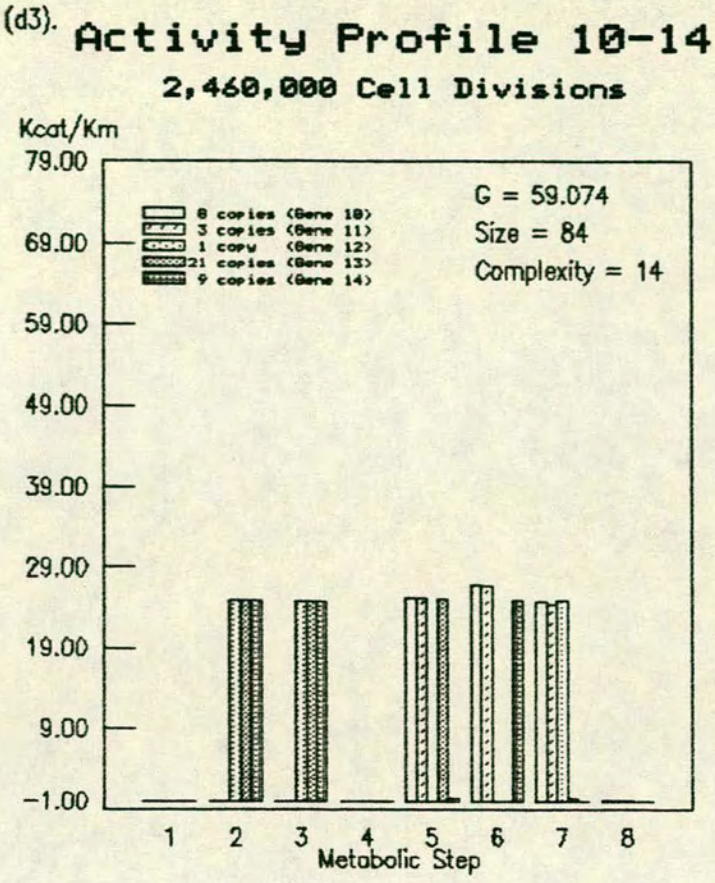

(d3).
Activity Profile 10-14
2,460,000 Cell Divisions
Kcat/Km
79.00
69.00
59.00
49.00
39.00
29.00
19.00
9.00
-1.00
8 copies (Gene 10)
3 copies (Gene 11)
1 copy (Gene 12)
21 copies (Gene 13)
9 copies (Gene 14)
G = 59.074
Size = 84
Complexity = 14
1 2 3 4 5 6 7 8
Metabolic Step

Figure 6.6.3

population to four significant figures. The activity profile had not changed in the final session.

Has the population found a stable set of proto-enzyme classes to which it is committed? Before directly addressing this question, it might be of interest to summarise the mean properties of the cells for the populations at the end of the four sessions:

Cell Divisions	*Activity/Enzyme*		*Growth Rate*
	Number	V_{max}/K_M	
360,000	4.703	14.427	52.28
1,020,000	4.391	17.132	58.66
1,710,000	4.341	17.383	58.88
2,460,000	4.361	17.383	59.07
Monofunctional	1.000	75.000	58.59

The above shows the number of cell divisions, the mean number of activities per enzyme, what the mean non-zero activity was, and the mean growth rate. The values for a cell with eight monofunctional enzymes are included for comparison. The cells at the end of the last three sessions have similar gross characteristics. Their average proto-enzymes have similar numbers of activities of similar quality, as figure 6.6.3 shows. Closer examination of individual cells, with the question of what classes of proto-enzymes can be identified, will reveal whether a single strategy has come to predominate in the population.

The classes of proto-enzyme identified will be labelled with lower-case Latin numerals, to distinguish them from those found with the former values of the Lennard-Jones constants.

Table 6.6.1

Class	*Activities*	*Representation of Class (No. of genes)*					
		360 (20)	*1020 (29)*	*1700 (64)*	*2460 (84)*	*2460f (80)*	*f×5 (392)*
i	2 4 6	6	0	0	0	0	0
ii	3 5	1	0	0	0	0	0
iii	1 4 6 7	2	1	6	10	11	54
iv	1 6 7 8	6	3	3	1	2	5
v	2 3 5	5	8	17	21	20	100
vi	1 4 8	0	7	16	22	19	95
vii	6 7 8	0	1	5	10	8	5
viii	5 6 7	0	3	8	11	10	48
ix	4 6 7	0	3	2	0	0	0
x	2 3 6	0	3	7	9	10	48

The columns are headed with the number of cell divisions (in thousands), and the size of the genome of the sampled cell. The last two columns of the table relate to the fastest cell found in the final population of the experiment. This cell (column headed 2460f) is included to allow some estimate of what the optimum allocation of protein among the classes might be (see below). It does seem as though class ix has been eliminated during the final session. Although polymorphism cannot be completely excluded, none of the cells of the final population whose genomes were reported possessed a gene coding for such a proto-enzyme.

Three cells drawn from the final population are shown in table 6.6.2. They comprise the fastest and slowest cells, and one other, arbitrarily chosen. It will be seen that the genome sizes and complexities are different in each case, but that the same proto-enzyme classes are present in each. While there are no favourable gene duplications or deletions in the fastest cell, this is not

so for the other two, though even after a series of locally best gene copy changes, the other cells are slower than the fastest cell drawn from the population. Note, incidently, that activities of less than 1.0 have been ignored in specifying the classes in table 6.6.1 (compare the description of the fastest cell in the two tables). Different proto-enzymes of the same class do not seem to differ in respect of which metabolic steps they display these low activities towards, as they are structurally very similar.

The last column of table 6.6.1 shows the result of the operation of multiplying each of the gene dosages of the fastest cell by five and selecting the gene duplication or deletion giving the highest growth rate until there are no further such events which yield higher rates. This exercise is shown in table 6.6.3, which shows that over 160 duplications and deletions occurred before a distribution that could not be improved by any single duplication or deletion event was reached. The gain in growth rate, from 59.13 to 59.23, seems meagre after such investment in the procedure. No proto-enzyme class is eliminated, though three genes are completely deleted. The redistribution of protein is quite considerable as a comparison of the last two columns of table 6.6.1 will show (classes vii and iv are the ones to examine).

There are, then, seven classes of proto-enzyme present in the final population. All but two of them are shown in table 6.6.1 as having three significant activities. The two exceptions (classes iii and iv) each have one low activity between 2.0 and 3.0, and three between 20.0 and 30.0, the latter being the range of the activities for the other classes also. The lower mean activity results from the statistics taking account of the very low activities ignored

312 ⇒

Table 6.6.2

FASTEST CELL SAMPLED AFTER 2,460,000 CELL DIVISIONS
EXPERIMENT 5

Genome Size = 80
Genome Complexity = 11
Growth Rate = 59.13119

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	7	24.10	0.00	0.00	24.84	0.00	0.00	0.00	26.15
2	1	24.10	0.00	0.00	24.83	0.00	0.00	0.00	26.79
3	11	24.10	0.00	0.00	24.82	0.00	0.00	0.00	26.69
4	4	0.00	0.00	0.00	0.21	0.00	26.86	24.75	24.77
5	9	0.00	0.00	0.00	0.07	25.27	26.86	24.75	0.16
6	11	24.04	0.00	0.00	25.05	0.00	2.27	24.57	0.00
7	4	0.00	0.00	0.00	0.21	0.00	26.79	24.46	23.90
8	2	24.03	0.00	0.00	0.20	0.00	2.26	24.43	24.96
9	1	0.00	0.00	0.00	0.07	25.27	26.79	24.45	0.16
10	20	0.00	24.94	24.74	0.01	25.09	0.00	0.51	0.03
11	10	0.00	24.94	24.74	0.00	0.46	24.96	0.03	0.01
V_{max}/K_M		9.63	9.35	9.28	9.38	9.49	9.53	9.67	9.38

EFFECT OF DUPLICATING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	9.81	9.24	9.16	9.57	9.37	9.41	9.55	9.59	59.11124
2	9.81	9.24	9.16	9.57	9.37	9.41	9.55	9.60	59.11715
3	9.81	9.24	9.16	9.57	9.37	9.41	9.55	9.59	59.11618
4	9.51	9.24	9.16	9.27	9.37	9.74	9.86	9.57	59.11534
5	9.51	9.24	9.16	9.26	9.68	9.74	9.86	9.27	59.11472
6	9.81	9.24	9.16	9.57	9.37	9.44	9.85	9.26	59.10599
7	9.51	9.24	9.16	9.27	9.37	9.74	9.85	9.56	59.10399
8	9.81	9.24	9.16	9.27	9.37	9.44	9.85	9.57	59.10566
9	9.51	9.24	9.16	9.26	9.68	9.74	9.85	9.27	59.11147
10	9.51	9.54	9.47	9.26	9.68	9.41	9.56	9.26	59.12639
11	9.51	9.54	9.47	9.26	9.38	9.72	9.55	9.26	59.12305

EFFECT OF DELETING NAMED GENE: V_{max}/K_M AND GROWTH RATE

Gene	V_{max}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8	Growth Rate
1	9.45	9.47	9.40	9.18	9.61	9.65	9.79	9.17	59.12080
2	9.45	9.47	9.40	9.18	9.61	9.65	9.79	9.16	59.11415
3	9.45	9.47	9.40	9.18	9.61	9.65	9.79	9.16	59.11525
4	9.75	9.47	9.40	9.49	9.61	9.31	9.48	9.19	59.11619
5	9.75	9.47	9.40	9.50	9.29	9.31	9.48	9.50	59.11671
6	9.45	9.47	9.40	9.18	9.61	9.62	9.48	9.50	59.12907
7	9.75	9.47	9.40	9.49	9.61	9.31	9.48	9.20	59.12880
8	9.45	9.47	9.40	9.49	9.61	9.62	9.48	9.18	59.12969
9	9.75	9.47	9.40	9.50	9.29	9.31	9.48	9.50	59.12032
10	9.75	9.15	9.08	9.50	9.29	9.65	9.79	9.50	59.10523
11	9.75	9.15	9.08	9.50	9.60	9.33	9.79	9.50	59.10884

SLOWEST CELL SAMPLED AFTER 2,460,000 CELL DIVISIONS
OF EXPERIMENT 5

Genome Size = 81
Genome Complexity = 12
Growth Rate = 59.04341

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	7	24.10	0.00	0.00	24.84	0.00	0.00	0.00	26.15
2	1	24.10	0.00	0.00	24.83	0.00	0.00	0.00	26.79
3	11	24.10	0.00	0.00	24.82	0.00	0.00	0.00	26.69
4	1	24.10	0.00	0.00	25.11	0.00	0.00	0.00	25.59
5	3	0.00	0.00	0.00	0.21	0.00	26.86	24.75	24.77
6	10	0.00	0.00	0.00	0.07	25.27	26.86	24.75	0.16
7	10	24.04	0.00	0.00	25.05	0.00	2.27	24.57	0.00
8	5	0.00	0.00	0.00	0.21	0.00	26.79	24.46	23.90
9	2	24.03	0.00	0.00	0.20	0.00	2.26	24.43	24.96
10	1	0.00	0.00	0.00	0.07	25.27	26.79	24.45	0.16
11	18	0.00	24.94	24.74	0.01	25.09	0.00	0.51	0.03
12	12	0.00	24.94	24.74	0.00	0.46	24.96	0.03	0.01
V_{max}/K_M		9.51	9.24	9.16	9.26	9.07	10.33	9.54	9.57

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
80	12	Delete	8	59.08473	11;	5 9 10 12
79	12	Delete	8	59.09771	11;	5 12
78	12	Delete	12	59.10736	9;	
77	11	Delete	4	59.12151	11;	1 2 3 7
77	11	None	0	59.12151		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 77
Genome Complexity = 11
Growth Rate = 59.12151

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	7	24.10	0.00	0.00	24.84	0.00	0.00	0.00	26.15
2	1	24.10	0.00	0.00	24.83	0.00	0.00	0.00	26.79
3	11	24.10	0.00	0.00	24.82	0.00	0.00	0.00	26.69
4	0								
5	3	0.00	0.00	0.00	0.21	0.00	26.86	24.75	24.77
6	10	0.00	0.00	0.00	0.07	25.27	26.86	24.75	0.16
7	10	24.04	0.00	0.00	25.05	0.00	2.27	24.57	0.00
8	3	0.00	0.00	0.00	0.21	0.00	26.79	24.46	23.90
9	2	24.03	0.00	0.00	0.20	0.00	2.26	24.43	24.96
10	1	0.00	0.00	0.00	0.07	25.27	26.79	24.45	0.16
11	18	0.00	24.94	24.74	0.01	25.09	0.00	0.51	0.03
12	11	0.00	24.94	24.74	0.00	0.46	24.96	0.03	0.01
V_{max}/K_M		9.69	9.39	9.32	9.41	9.54	9.85	9.40	9.11

CELL SAMPLED AFTER 2,460,000 CELL DIVISIONS
OF EXPERIMENT 5

Genome Size = 92
Genome Complexity = 13
Growth Rate = 59.07649

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	6	24.10	0.00	0.00	24.84	0.00	0.00	0.00	26.15
2	2	24.10	0.00	0.00	24.83	0.00	0.00	0.00	26.79
3	11	24.10	0.00	0.00	24.82	0.00	0.00	0.00	26.69
4	2	24.10	0.00	0.00	25.11	0.00	0.00	0.00	25.59
5	2	24.04	0.00	0.00	25.05	0.00	2.60	24.97	0.00
6	4	0.00	0.00	0.00	0.21	0.00	26.86	24.75	24.77
7	8	0.00	0.00	0.00	0.07	25.27	26.86	24.75	0.16
8	12	24.04	0.00	0.00	25.05	0.00	2.27	24.57	0.00
9	7	0.00	0.00	0.00	0.21	0.00	26.79	24.46	23.90
10	2	24.03	0.00	0.00	0.20	0.00	2.26	24.43	24.96
11	2	0.00	0.00	0.00	0.07	25.27	26.79	24.45	0.16
12	24	0.00	24.94	24.74	0.01	25.09	0.00	0.51	0.03
13	10	0.00	24.94	24.74	0.00	0.46	24.96	0.03	0.01
V_{max}/K_M		9.68	9.22	9.14	9.53	9.34	9.24	10.03	9.50

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
91	13	Delete	8	59.09936	12 13;	1 2 3 4 5 9 10
90	13	Delete	10	59.11337	12 13;	4 8 9
90	13	None	0	59.11337		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 90
Genome Complexity = 13
Growth Rate = 59.11337

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	6	24.10	0.00	0.00	24.84	0.00	0.00	0.00	26.15
2	2	24.10	0.00	0.00	24.83	0.00	0.00	0.00	26.79
3	11	24.10	0.00	0.00	24.82	0.00	0.00	0.00	26.69
4	2	24.10	0.00	0.00	25.11	0.00	0.00	0.00	25.59
5	2	24.04	0.00	0.00	25.05	0.00	2.60	24.97	0.00
6	4	0.00	0.00	0.00	0.21	0.00	26.86	24.75	24.77
7	8	0.00	0.00	0.00	0.07	25.27	26.86	24.75	0.16
8	11	24.04	0.00	0.00	25.05	0.00	2.27	24.57	0.00
9	7	0.00	0.00	0.00	0.21	0.00	26.79	24.46	23.90
10	1	24.03	0.00	0.00	0.20	0.00	2.26	24.43	24.96
11	2	0.00	0.00	0.00	0.07	25.27	26.79	24.45	0.16
12	24	0.00	24.94	24.74	0.01	25.09	0.00	0.51	0.03
13	10	0.00	24.94	24.74	0.00	0.46	24.96	0.03	0.01
V_{max}/K_M		9.36	9.42	9.35	9.46	9.55	9.39	9.71	9.43

Table 6.6.3

FASTEST CELL SAMPLED AFTER 2,460,000 CELL DIVISIONS
OF EXPERIMENT 5 WITH ALL GENE DOSAGES INCREASED FIVE-FOLD

Genome Size = 400
Genome Complexity = 11
Growth Rate = 59.13119

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	35	24.10	0.00	0.00	24.84	0.00	0.00	0.00	26.15
2	5	24.10	0.00	0.00	24.83	0.00	0.00	0.00	26.79
3	55	24.10	0.00	0.00	24.82	0.00	0.00	0.00	26.69
4	20	0.00	0.00	0.00	0.21	0.00	26.86	24.75	24.77
5	45	0.00	0.00	0.00	0.07	25.27	26.86	24.75	0.16
6	55	24.04	0.00	0.00	25.05	0.00	2.27	24.57	0.00
7	20	0.00	0.00	0.00	0.21	0.00	26.79	24.46	23.90
8	10	24.03	0.00	0.00	0.20	0.00	2.26	24.43	24.96
9	5	0.00	0.00	0.00	0.07	25.27	26.79	24.45	0.16
10	100	0.00	24.94	24.74	0.01	25.09	0.00	0.51	0.03
11	50	0.00	24.94	24.74	0.00	0.46	24.96	0.03	0.01
V_{max}/K_M		9.63	9.35	9.28	9.38	9.49	9.53	9.67	9.38

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
399	11	Delete	7	59.13318	10 11;	1 6 8 9
398	11	Delete	6	59.13510	4 10 11;	1 7 8
397	11	Delete	8	59.13605	2 4;	1 6 7
398	11	Duplicate	4	59.13697	2 3;	6 7
397	11	Delete	7	59.13849	2 3;	6 8 9
398	11	Duplicate	4	59.13938	2 3;	6 7
397	11	Delete	7	59.14093	2 3;	6 8 9
398	11	Duplicate	4	59.14180	2 3;	6 7
397	11	Delete	7	59.14337	2 3;	6 8 9
398	11	Duplicate	4	59.14421	2 3;	6 7
397	11	Delete	7	59.14581	2 3;	6 8 9
398	11	Duplicate	4	59.14662	2 3;	6 7
397	11	Delete	7	59.14825	2 3;	6 8 9
398	11	Duplicate	4	59.14904	2 3;	6 7
397	11	Delete	7	59.15069	2 3;	6 8 9
398	11	Duplicate	4	59.15145	2 3;	6 7
397	11	Delete	7	59.15312	2 3;	6 8 9
398	11	Duplicate	4	59.15386	2 3;	6 7
397	11	Delete	7	59.15556	2 3;	6 8 9
398	11	Duplicate	4	59.15627	2 3;	6 7 8

contd over page....

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
397	11	Delete	7	59.15800	2 3 10;	6 8 9
398	11	Duplicate	4	59.15868	2 3;	6 7 8
397	11	Delete	7	59.16043	2 3 10;	6 8 9
398	11	Duplicate	4	59.16109	2 3;	6 7 8
397	11	Delete	7	59.16287	2 3 10;	6 8 9
398	11	Duplicate	4	59.16350	2 3;	6 7 8
397	11	Delete	7	59.16530	2 3 10;	6 8 9
398	11	Duplicate	4	59.16591	2 3;	6 7 8
397	11	Delete	7	59.16773	2 3 10;	6 8 9
396	11	Delete	7	59.16834	2 3 4;	6 8
397	11	Duplicate	4	59.17017	2 3 5;	6 10
396	11	Delete	7	59.17080	2 3 4;	6 8
397	11	Duplicate	4	59.17260	2 3 5;	6 10
396	11	Delete	7	59.17326	2 3 4;	6 8
397	11	Duplicate	4	59.17503	2 3 5;	6 10
396	11	Delete	7	59.17572	2 3 4;	6 8
397	11	Duplicate	4	59.17746	2 3 5;	6 10
396	11	Delete	7	59.17817	2 3 4;	6 8
397	11	Duplicate	4	59.17989	2 5;	6 10
396	10	Delete	7	59.18063	2 3 4;	6 8
397	10	Duplicate	4	59.18232	2 5;	6 10
398	10	Duplicate	4	59.18274	2 3;	6 8
397	10	Delete	8	59.18363	2 3 10;	6 9
398	10	Duplicate	2	59.18444	3;	9 11
397	10	Delete	1	59.18490	10;	6 8
398	10	Duplicate	2	59.18569	3;	9 11
397	10	Delete	1	59.18616	10;	6 8
398	10	Duplicate	2	59.18694	3;	9 11
397	10	Delete	1	59.18743	10;	6 8
398	10	Duplicate	2	59.18820	3;	9 11
397	10	Delete	1	59.18870	10;	6 8
398	10	Duplicate	2	59.18945	3;	9 11
397	10	Delete	1	59.18996	10;	6 8
398	10	Duplicate	2	59.19070	3;	9 11
397	10	Delete	1	59.19123	10;	6 8
398	10	Duplicate	2	59.19195	3;	9 11
397	10	Delete	1	59.19250	10;	6 8
398	10	Duplicate	2	59.19320	3;	9 11
397	10	Delete	1	59.19376	10;	6 8
398	10	Duplicate	2	59.19445	3;	9 11
397	10	Delete	1	59.19503	10;	6 8
398	10	Duplicate	2	59.19570	3;	9 11
397	10	Delete	1	59.19629	10;	6 8
398	10	Duplicate	2	59.19695	3;	9 11
397	10	Delete	1	59.19755	10;	6 8
398	10	Duplicate	2	59.19820	3;	9 11
397	10	Delete	1	59.19882	10;	6 8

contd over page...

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
396	10	Delete	9	59.19945	2 3;	
397	10	Duplicate	4	59.19964	5;	1 11
396	10	Delete	8	59.20023	10;	1
395	10	Delete	11	59.20091	2 3;	9
394	10	Delete	1	59.20117		6 8 9
395	10	Duplicate	2	59.20217	3;	5 9 11
394	10	Delete	1	59.20244		6 8 9
395	10	Duplicate	2	59.20343	3;	5 9 11
394	10	Delete	1	59.20372		6 8 9
395	10	Duplicate	2	59.20468	3;	5 9 11
394	10	Delete	1	59.20499		6 8 9
395	10	Duplicate	2	59.20594	3;	5 9 11
394	10	Delete	1	59.20626		6 8 9
395	10	Duplicate	2	59.20719	3;	5 9 11
394	10	Delete	1	59.20753		6 8 9
395	10	Duplicate	2	59.20845	3;	5 9 11
394	10	Delete	1	59.20880		6 8 9
395	10	Duplicate	2	59.20970	3;	5 9 11
394	10	Delete	1	59.21007		6 8 9
395	10	Duplicate	2	59.21096	3;	5 9 11
394	10	Delete	1	59.21134		6 8 9
395	10	Duplicate	2	59.21221	3;	5 9 11
394	10	Delete	1	59.21261		6 8 9
395	10	Duplicate	2	59.21347	3;	5 9 11
394	10	Delete	1	59.21387		8 9
395	10	Duplicate	2	59.21472	3;	5 9 11
394	10	Delete	1	59.21514		8 9
393	10	Delete	9	59.21598	2 3;	5 11
392	10	Delete	11	59.21621	2;	
391	10	Delete	1	59.21689		6 8
392	10	Duplicate	2	59.21747	3;	9
391	10	Delete	1	59.21816		6 8
392	10	Duplicate	2	59.21873	3;	9
391	10	Delete	1	59.21944		6 8
392	10	Duplicate	2	59.21999	3;	9
391	10	Delete	1	59.22072		6 8
392	10	Duplicate	2	59.22125	3;	9
391	10	Delete	1	59.22199		6 8
392	10	Duplicate	2	59.22251	3;	8 9
391	10	Delete	1	59.22327		6 8
392	10	Duplicate	2	59.22377	3;	8 9
391	10	Delete	1	59.22454		6 8
392	10	Duplicate	2	59.22503	3;	8 9
391	10	Delete	1	59.22581		6 8
392	10	Duplicate	2	59.22629	3;	8 9
391	10	Delete	1	59.22709		6 8

contd over page...

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
392	10	Duplicate	2	59.22755	3;	8 9
391	10	Delete	1	59.22836		6 8
392	10	Duplicate	2	59.22880	3;	8 9
391	9	Delete	1	59.22963		6 8
392	9	Duplicate	2	59.23006	3;	8 9
391	9	Delete	8	59.23074		6
392	9	Duplicate	4	59.23095		
391	9	Delete	9	59.23122		8
392	9	Duplicate	5	59.23166		8
391	9	Delete	9	59.23194		8
392	9	Duplicate	5	59.23237		8
391	8	Delete	9	59.23266		8
392	8	Duplicate	5	59.23308		8
391	8	Delete	8	59.23321		
392	8	Duplicate	2	59.23356	3;	11
392	8	None	0	59.23356		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 392
Genome Complexity = 8
Growth Rate = 59.23356

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	0								
2	40	24.10	0.00	0.00	24.83	0.00	0.00	0.00	26.79
3	55	24.10	0.00	0.00	24.82	0.00	0.00	0.00	26.69
4	42	0.00	0.00	0.00	0.21	0.00	26.86	24.75	24.77
5	48	0.00	0.00	0.00	0.07	25.27	26.86	24.75	0.16
6	54	24.04	0.00	0.00	25.05	0.00	2.27	24.57	0.00
7	0								
8	5	24.03	0.00	0.00	0.20	0.00	2.26	24.43	24.96
9	0								
10	100	0.00	24.94	24.74	0.01	25.09	0.00	0.51	0.03
11	48	0.00	24.94	24.74	0.00	0.46	24.96	0.03	0.01
V_{max}/K_M		9.46	9.42	9.34	9.50	9.55	9.56	9.51	9.48

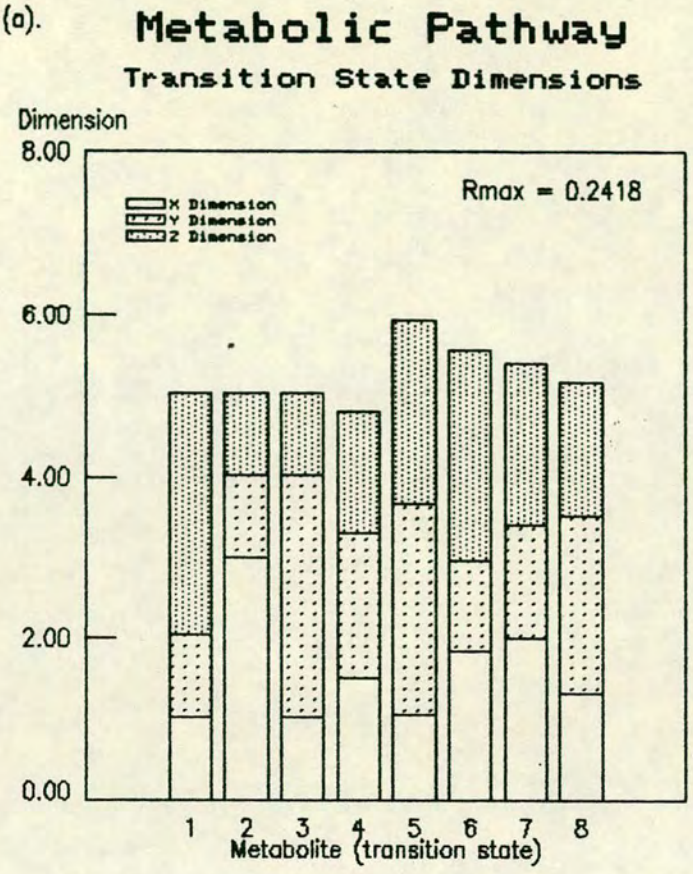
completely in the construction of the table 6.6.1, and often not visible in the activity profile charts. It seems that a value in the mid-twenties is standard for a proto-enzyme activity in the present scenario.

What is the explanation for the universality of three activities per proto-enzyme, and the relative invariance of the quality of these activities? There is an obvious hypothesis. There are three axes, each of which contribute equally to the 'activation energy' of k_{cat}/K_M. The maximum possible specificity constant is 75, one third of which is 25. The enzyme has three axes to tinker with, and eight potential substrates. The independence of the axes means that it is by no means pre-ordained that the cell should happen to 'choose' the same substrate for each axis. Three axes, three substrates. The additional low activities are fortuitous, resulting from the particular set of active-site axes not excluding them. The procedure that was defined previously, to calculate a 'perfect proto-enzyme' for any given set of substrates, used this independence of the three axes.

Looking at the sequences of the proto-enzymes will reveal whether this hypothesis is correct, for if it is, the activities of each catalyst for each substrate derive largely from interaction with a single axis of the proto-enzyme active site.

Consider the *x*-axes of the transition states (figure 6.6.4a) and proto-enzymes (figure 6.6.4b). There are six distinct values for the *x*-axes of the eight transition states (1, 3, and 5 are the same). There are four such values for the ten proto-enzyme classes (iii, iv, vii, viii, and ix are the same, as are i,

(a).



(b).

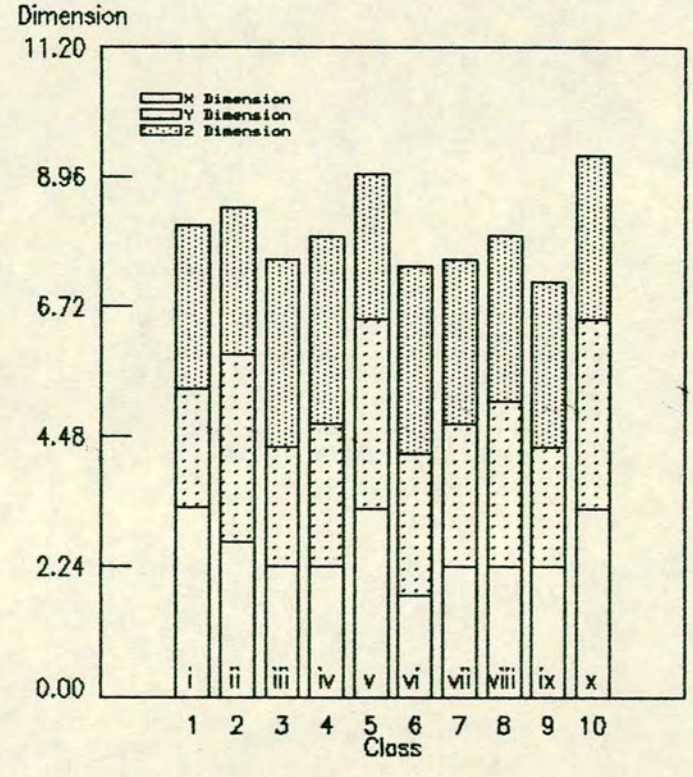


Figure 6.6.4 *Experiment 5* 313

v, and x). The first thing that strikes one when looking at the two figures is that there is no catalyst which has exploited the three identical transition state *x*-axes, which represent, incidently, the smallest *x*-axes of the reaction intermediates of the pathway. The smallest *x*-axis of the proto-enzyme active sites is found in class vi proto-enzymes, which have activity not to 1, 3, and 5 but to 1, 4, and 8. The *x*-axis must be larger, therefore, than that appropriate for maximum utilisation of binding energy in the formation of transition state 1 (and 8). At best, it could be tailored for maximum interaction with transition state 4. However, the other two axes could not be so tailored without excluding reactions 1 and 8 altogether.

Where several proto-enzyme sites have the same dimensions in one or more axes, they are likely to have a reaction in common. In fact, classes iii, iv, vii, viii and ix, which have the same *x*-axis dimension, share two common substrates, 6 and 7. However, in classes iii and iv, the activity for step 6 is low (around 2), while in the others it is higher than 25 (you guessed: 27), suggesting perhaps that the common axis dimension makes a very modest contribution to the reaction, which accounts for the low activity for two of the classes, and the high activity for the others (where some other axis makes the major contribution). This kind of argument seems very plausible. A breakdown of the contributions of the various axes to the different activities is given in table 6.6.4.

Table 6.6.4

Class	*Axis*	*Activity Contribution (per metabolic step)* 1	2	3	4	5	6	7	8
i	x	-	24.49	-	0.00	-	0.00	0.00	-
	y	-	0.01	*	24.37	*	0.02	0.13	*
	z	*	0.00	-	0.00	-	24.83	0.03	-
ii	x	-	*	0.00	0.00	0.00	-	0.13	0.00
	y	-	-	24.74	0.00	0.14	-	0.00	0.01
	z	*	-	0.00	0.01	24.95	*	0.00	0.02
iii	x	0.00	*	-	0.06	-	2.10	24.42	-
	y	0.01	-	*	24.99	*	0.02	0.14	*
	z	24.03	-	-	0.00	-	0.15	0.00	-
iv	x	0.00	*	-	0.06	-	2.10	24.42	0.02
	y	0.00	-	*	0.15	*	0.00	0.01	24.95
	z	24.03	-	-	0.00	-	0.15	0.00	0.00
v	x	-	24.94	0.00	0.00	0.00	-	0.00	0.00
	y	-	0.00	24.74	0.00	0.14	-	0.00	0.01
	z	*	0.00	0.00	0.00	24.95	*	0.50	0.22
vi	x	0.00	*	-	24.69	-	*	*	1.85
	y	0.00	-	*	0.15	*	-	-	24.94
	z	24.03	-	-	0.00	-	-	-	0.00
vii	x	-	*	-	0.06	-	2.17	24.71	0.02
	y	-	-	*	0.15	*	0.00	0.01	24.75
	z	*	-	-	0.00	-	24.68	0.03	0.00
viii	x	-	*	-	0.06	0.00	2.17	24.71	0.02
	y	-	-	*	0.01	24.96	0.00	0.00	0.14
	z	*	-	-	0.00	0.30	24.68	0.03	0.00
ix	x	-	*	-	0.06	-	2.10	24.42	-
	y	-	-	*	23.59	*	0.02	0.13	*
	z	*	-	-	0.00	-	24.99	0.03	-
x	x	-	24.94	0.00	0.00	0.00	0.00	0.00	0.00
	y	-	0.00	24.74	0.00	0.14	0.00	0.00	0.01
	z	*	0.00	0.00	0.00	0.32	24.96	0.03	0.00

The asterisks in the above table mark the axis (there is only ever one in this set of data) which excludes the particular reaction from the active site.

The table clearly reveals the correctness of the hypothesis proposed above, that each of the three activities of a proto-enzyme class derive very largely from one axis interaction. Indeed, with these values of the Lennard-Jones constants, it turns out that activities of this value could not have been generally obtained in any other way (as with the current set of reaction intermediates, there is no possible axis dimension that could display activities of around one-third maximum towards three differently sized species: the differences in size are too large).

Each reaction is catalysed by at least three different classes in the table (remember, though, that only seven of the ten classes are present in the final population). Interestingly, not only is a single axis always largely responsible for a given activity, but where the same reaction is catalysed by more than one proto-enzyme class, it is, with only a single exception, the same active-site axis that is responsible (the exception is the catalysis of reaction 5 by class viii, which is largely due to the *y*-axis of the active site, where the *z*-axis is responsible in class v, and in the pathological [see next paragraph] two-substrate class ii).

As identical axis dimensions in different substrates are found in this pathway, it would seem that the present population could have done better by evolving proto-enzyme active sites tailored for sets of substrates sharing common axes. The class ii proto-enzymes, which were rejected early in the experiment, could have done this, as the two substrates of that class (3 and 5) have identical *x*-axes. However, the *x*-axis of the proto-enzyme was very

much larger than required for maximum binding, and the proto-enzyme was presumably lost before suitable mutations occurred to improve the fit.

The above observations are a consequence of the lack of a notion of 'overall fit' in the model as it stands. The concept of complementarity is in this sense primitive. The strategy of independently evolving the three axes for independent substrates is likely to be advantageous in most scenarios compared with the monofunctional solution, because of the small additional activities for steps other than the proto-enzymes 'triplet'.

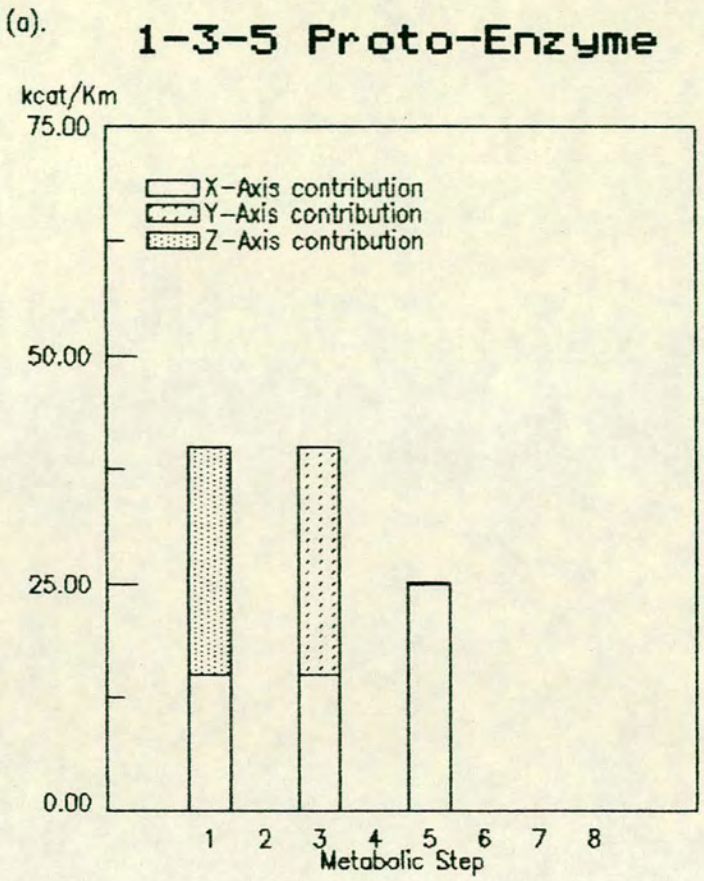
That there is no advantage in the present model for evolving monofunctionality does not adversely affect the objective of the present work to use a multi-enzyme systems model for a brute-force approach to evolutionary phenomena. Indeed, in the present series of experiments, a pathway favouring multifunctionality was chosen (after the preliminary experiments using randomly generated pathways were examined) because of the probable 'ruggedness' of the adaptive landscape.

Under the model, for a given scenario in which monofunctionality is physically realisable, the best monofunctional proto-enzyme has the theoretical maximum activity for its substrate. A cell comprising only such enzymes will therefore contain a set of catalysts with identical levels of activity. The optimum distribution of protein among these catalysts is simply equal (the molecular weights, it will be recalled, are hidden inside the catalytic coefficients). To put it succinctly, each proto-enzyme should have the same number of genes.

Things are not so simple in the multifunctional case. The proto-enzymes have different numbers of activities (three major ones, but additional small activities are often present) and even where the numbers are the same, the levels of activity are not equal (though for the present scenario they are generally similar).

In the present pathway, there are reaction intermediates which have one or more identical or very similar axis dimensions. We can use the earlier defined method for obtaining 'perfect' proto-enzymes (more effective in the present scenario, where the average difference in size of the corresponding axes of different reaction intermediates is large compared with R_{max}) to obtain the characteristics of proto-enzymes which exploit these identical/similar axes. There are two such sets of axes, and the profiles of the three-substrate proto-enzymes which encompass them are shown in figure 6.6.5. Cells containing these classes are expected to be at an advantage. No such cells evolved in the present experiment though classes corresponding to them did evolve in some of the earlier experiments with the lower Lennard-Jones *B* constant. It is therefore an interesting exercise to add a single copy gene for each of these proto-enzymes to the genome of the fsatest cell at the end of the experiment and ask what effect on growth rate there is, and how the allocation of protein is affected.

This is done in table 6.6.5, which reveals that the growth rate of the cell is indeed improved. In fact, the growth rate is greater than in the cell with all genes increased in dosage five-fold, and put through the duplication/deletion procedure (recall table 6.6.3). In the newly formed cell with these two



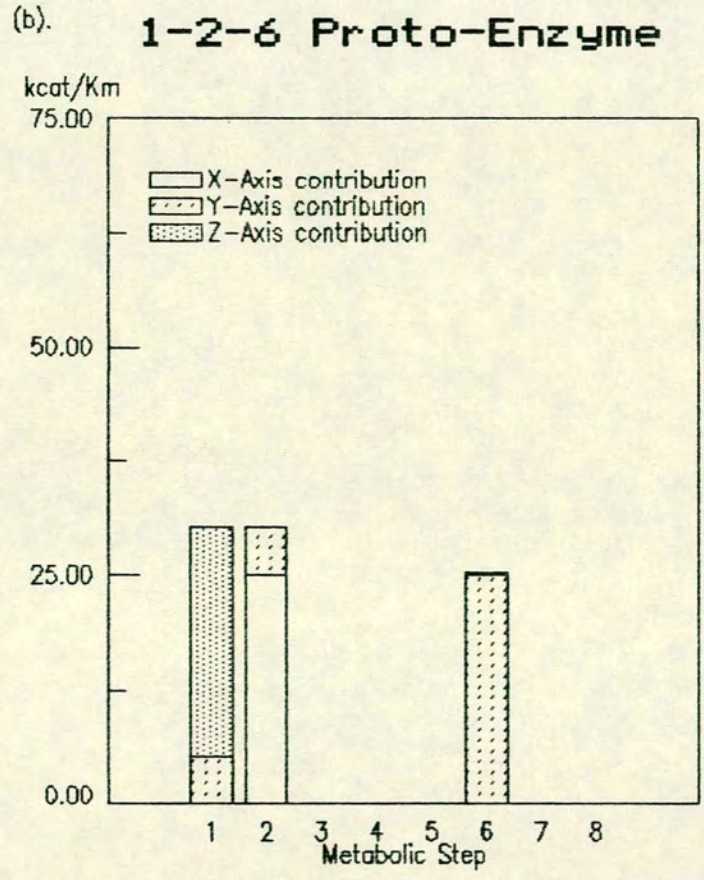


Figure 6.6.5 *Experiment 5*

additional genes, duplication of either of the newly added genes is favourable. So also is duplication of gene 4. With the addition of the new genes, the poorest activities are steps 4 and 8, while the highest activity is that towards step 1. The only proto-enzyme class of the final population which has activity for either step 4 or 8 but not for step 1 is class vii (table 6.6.1), and this is why duplication of gene 4, which belongs to class vii, is favourable.

Recall that there were no favourable changes in gene copy number in the original fastest cell. When its genome size was increased five-fold and put through the repeated duplication/deletion procedure a large number of duplications and deletions occurred, with genes 7 and 9 being deleted completely (table 6.6.3). These genes were less active sisters of genes 4 and 5 respectively. Addition of the two new genes to the original cell produces a cell with favourable single duplications. Application of the duplication/deletion procedure eliminated the same two genes, and also gene 8, the sole representative of class iv in the cell, while the newly added genes were each duplicated several times. After the procedure, which involved some fifty duplications and deletions (table 6.6.5) the final cell had a growth rate exceeding 60, and had increased its genome size by one copy (while decreasing its genome complexity by three genes as already stated). It seems that the evolution of proto-enzymes which exploit the similarity of the reaction intermediates of the pathway would have been favoured, and presumably did not occur because of the unavailability of suitable mutations. None of the genes present in the final population can mutate in a single step to either of the new genes. Such a change is possible in two steps (for each gene), without

producing null intermediates, presumably the first mutational step of these is either disadvantageous in the particular genetic background of the population, or has not occurred.

Table 6.6.5

FASTEST CELL SAMPLED AFTER 2,460,000 CELL DIVISIONS
OF EXPERIMENT 5 WITH TWO ADDITIONAL SINGLE-COPY GENES
CODING FOR 'PERFECT' 1-3-5 & 1-2-6 ENZYMES

Genome Size = 82
Genome Complexity = 13
Growth Rate = 59.45037

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	7	24.10	0.00	0.00	24.84	0.00	0.00	0.00	26.15
2	1	24.10	0.00	0.00	24.83	0.00	0.00	0.00	26.79
3	11	24.10	0.00	0.00	24.82	0.00	0.00	0.00	26.69
4	4	0.00	0.00	0.00	0.21	0.00	26.86	24.75	24.77
5	9	0.00	0.00	0.00	0.07	25.27	26.86	24.75	0.16
6	11	24.04	0.00	0.00	25.05	0.00	2.27	24.57	0.00
7	4	0.00	0.00	0.00	0.21	0.00	26.79	24.46	23.90
8	2	24.03	0.00	0.00	0.20	0.00	2.26	24.43	24.96
9	1	0.00	0.00	0.00	0.07	25.27	26.79	24.45	0.16
10	20	0.00	24.94	24.74	0.01	25.09	0.00	0.51	0.03
11	10	0.00	24.94	24.74	0.00	0.46	24.96	0.03	0.01
12	1	39.98	0.00	39.98	0.00	25.16	0.00	0.00	0.00
13	1	30.16	30.16	0.00	0.00	0.00	25.14	0.00	0.00
V_{max}/K_M		10.25	9.49	9.54	9.15	9.56	9.60	9.43	9.15

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
83	13	Duplicate	12	59.64024	4 13;	
84	13	Duplicate	12	59.77643	4 7 13;	10 11
85	13	Duplicate	12	59.86492	4 5 7 13;	10 11
86	13	Duplicate	4	59.93079	5 7 9 12 13;	10 11
87	13	Duplicate	12	59.99471	4 7 13;	8 10 11
88	13	Duplicate	4	60.04765	2 3 7 12 13;	8 10 11
89	13	Duplicate	12	60.08973	2 3 4 7 13;	8 10 11
90	13	Duplicate	4	60.13069	1 2 3 7 12 13;	8 10 11
91	13	Duplicate	13	60.16968	1 2 3 4 7 12;	8 9 10 11
90	13	Delete	8	60.19560	1 2 3 4 6 7 12 13;	9 10 11
89	13	Delete	10	60.23466	1 2 3 4 6 7 12 13;	8 11
90	13	Duplicate	12	60.26500	1 2 3 4 7 13;	8 9 10 11
91	13	Duplicate	4	60.29790	1 2 3 7 13;	8 10 11
90	12	Delete	8	60.32741	1 2 3 4 12 13;	5 9 10 11
89	12	Delete	11	60.35830	1 2 3 4 6 7 12 13;	5 9 10
90	12	Duplicate	13	60.38329	2 3 4 7 12;	5 9 10 11
89	12	Delete	11	60.40870	1 2 3 4 7 12;	5 9 10
90	12	Duplicate	4	60.42812	7 12 13;	5 9 10 11
89	11	Delete	9	60.45214	12 13;	5
90	11	Duplicate	12	60.47227	4 13;	11
91	11	Duplicate	4	60.49305	7 13;	10 11
92	11	Duplicate	13	60.51309	12;	5

contd next page....

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
91	11	Delete	11	60.52895	4 12;	5
92	11	Duplicate	13	60.53706	12;	5
93	11	Duplicate	12	60.54858	4;	5 11
94	11	Duplicate	4	60.56314	7;	10 11
93	11	Delete	5	60.57351		11
92	11	Delete	11	60.58368	4;	
91	11	Delete	1	60.59404	13;	2 3
90	11	Delete	5	60.59801		11
91	11	Duplicate	4	60.60778		11
90	11	Delete	7	60.60935		11
91	11	Duplicate	4	60.61860		11
90	11	Delete	7	60.62068		11
91	11	Duplicate	4	60.62940		11
90	11	Delete	7	60.63199		11
91	11	Duplicate	4	60.64020		1 11
90	10	Delete	7	60.64329		11
91	10	Duplicate	4	60.65098		1 11
90	10	Delete	11	60.65144		
89	10	Delete	1	60.66574	10 13;	2 3
88	10	Delete	5	60.67486	13;	
87	10	Delete	1	60.67720		
86	10	Delete	11	60.68561	2 3 6;	10
87	10	Duplicate	13	60.69437		5
86	10	Delete	11	60.69707		
85	10	Delete	5	60.69755		
84	10	Delete	1	60.70873		2 3
83	10	Delete	10	60.70975		
83	10	None	0	60.70975		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 83
Genome Complexity = 10
Growth Rate = 60.70975

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	3	24.10	0.00	0.00	24.84	0.00	0.00	0.00	26.15
2	1	24.10	0.00	0.00	24.83	0.00	0.00	0.00	26.79
3	11	24.10	0.00	0.00	24.82	0.00	0.00	0.00	26.69
4	16	0.00	0.00	0.00	0.21	0.00	26.86	24.75	24.77
5	5	0.00	0.00	0.00	0.07	25.27	26.86	24.75	0.16
6	11	24.04	0.00	0.00	25.05	0.00	2.27	24.57	0.00
7	0								
8	0								
9	0								
10	18	0.00	24.94	24.74	0.01	25.09	0.00	0.51	0.03
11	3	0.00	24.94	24.74	0.00	0.46	24.96	0.03	0.01
12	9	39.98	0.00	39.98	0.00	25.16	0.00	0.00	0.00
13	6	30.16	30.16	0.00	0.00	0.00	25.14	0.00	0.00
V_{max}/K_M		14.06	8.49	10.60	7.85	9.71	9.82	9.63	9.60

Scenario Favouring Monofunctionality

In the experiments described so far, it has always been possible to evolve a cell containing multifunctional proto-enzymes which would grow faster than any cell containing only monofunctional ones. In the first scenario, used in experiments 1 to 3 (and in the introductory and dispersed-storage 'A' experiments), the advantage of multifunctionality was highly significant, with final populations in some cases comprised of cells with growth rates twenty-five per cent (or more) higher than achievable by a specialist monofunctional cell. Even when the cells were 'trapped' in a two- or three-gene complexity genome, the growth rates exceeded that of the monofunctional state.

In experiments 4 and 5 the high Lennard-Jones *B* constant reduced the potential magnitude of these advantages considerably. Indeed, my intuition at the time of devising the experiment was that multifunctional cells would have a small advantage solely because of those sets of transition state axes dimensions which were very similar in different metabolic steps. In fact, exploitation of these similarities failed to evolve in the two experiments (the first of which, of course, was abruptly terminated by extinction), yet the growth rates of the cells of experiment 5 passed the monofunctional maximum, nevertheless, though by a much smaller margin than in first-scenario experiments. Only at this point did I appreciate what was, in retrospect, quite obvious from the earlier experiments: the independence of the axes contribution to the catalytic coefficient guaranteed that multifunctionality would evolve. Even if there were no 'similarities' in the substrate dimensions (that is, no axis could contribute to more than a single activity), so that the maximum multifunctional

and monofunctional growth rates were the same, multifunctionality would almost inevitably be adopted, because there are many more multifunctional 'solutions' than monofunctional ones. Note, however, that not all multifunctional solutions, with all axis contributions at maximum, would be equivalent, because the distribution of activities to the different steps should ideally (as the maximum is the same in each case) be equal if growth rate is to be maximal. This is 'easy' to achieve in any viable monofunctional cell by changes in allocation, but is harder for a multifunctional cell because duplication and deletion of genes affects several activities simultaneously.

The present experiment captures the notion of complementarity or overall fit by adding the square root of the product of the contributions of the three binding interactions to their sum to obtain the total binding energy (see equation 3.4):

$$E_s = 2\ [E_x + E_y + E_z + (E_x \times E_y \times E_z)^{1/2}]$$

The catalytic coefficient, recall, is directly proportional to this.

Figure 6.7.1 shows the shape of this function for a cubic active-site interacting with a cubic substrate (i.e. the case where $E_x = E_y = E_z$). One important feature which the curve does not show, however, because it considers only the cubic case, is that significant additions are only possible if all three axes make at least moderate contributions to the activity. An interaction energy considerably less than unity in any one axis can limit the product of the three energies to a low value, even with maximum contributions from the other axes.

Experiment 6;

Figure 6.7.1

The graph displays the catalytic activity of a proto-enzyme towards a substrate with equal contact separation for each of the three axes of interaction.

The difference between the two curves represents the additional activity due to adding a term for 'overall fit'.

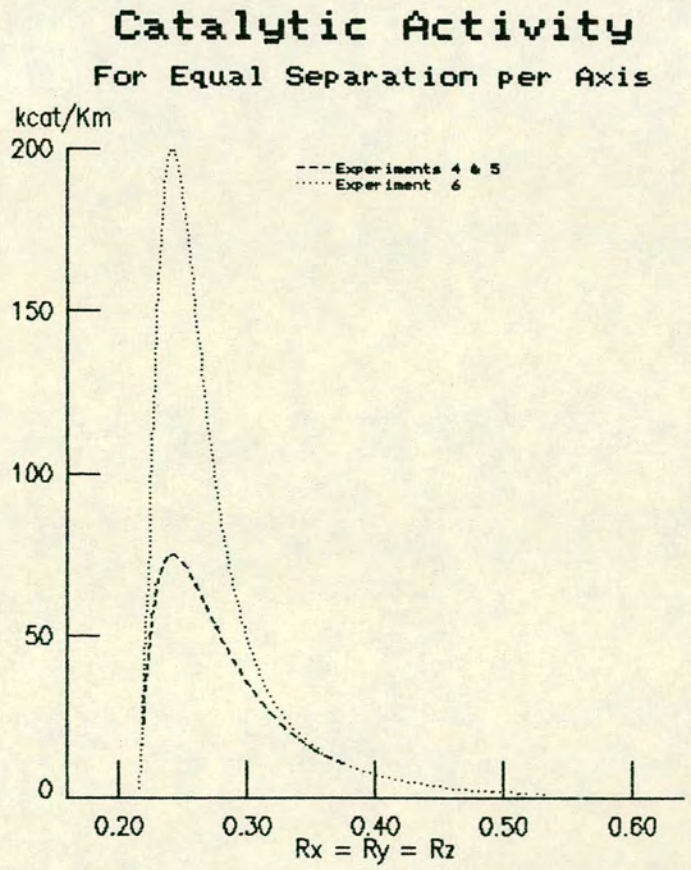


Figure 6.7.1

Experiment 6

It turns out that the function shown in figure 6.7.1 produces a landscape in which sequences must fall very close to the highest activity form to be captured in its "gravitational pull". In the course of the seven and a half million cell divisions of experiment 6 (figure 6.7.2a) a single monofunctional proto-enzyme does evolve (figure 6.7.4c), but the mean growth rate passes that of experiment 5 by only a small margin (compare figure 6.6.1b on page 289, with figure 6.7.2b). The theoretically possible growth rate for a cell with eight maximum-activity monofunctional protoenzymes, 156.4, is not approached.

This non-realisation of the system's full potential is not due to the cells of the population falling into a small-genome trap, in which any mutation giving rise to a monofunctional proto-enzyme would have been lethal, for the average genome size settles firmly at eleven after some two million cell divisions (figures 6.7.1c and 6.7.1e). Neither is it due to local forces favouring multifunctionality because of similarities in the reaction species for different steps, as the pathway used in the experiment, shown in figure 6.7.3a, has no similar pair of transition state dimensions in any axis that can be exploited to favour multifunctional proto-enzymes.

The single monofunctional proto-enzyme that did arise, with almost maximum activity for the fifth step, appeared after six hundred and ninety thousand cell divisions. Its ancestor, in which two of the three axes of the transition state for reaction five are tightly bound, first appeared at two hundred and ten thousand cell divisions, and had a k_{cat}/K_M for the fifth step of 49.12,

=> *331*

Experiment 6;

Figure **6.7.2**

a) Population size – reduced to 100 at end of each session

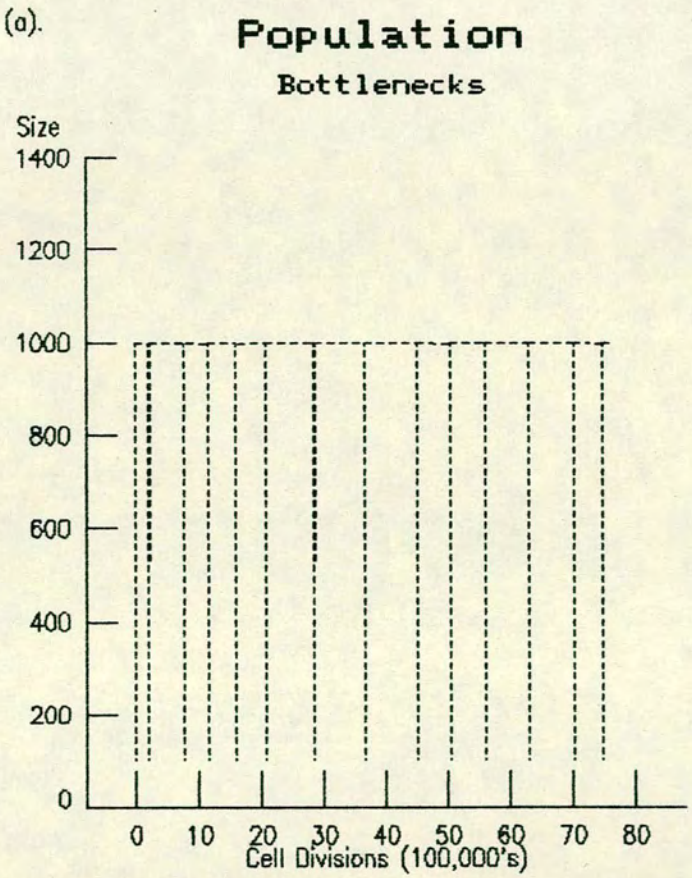
b) Progress of growth rate

c) Mean genome size & complexity

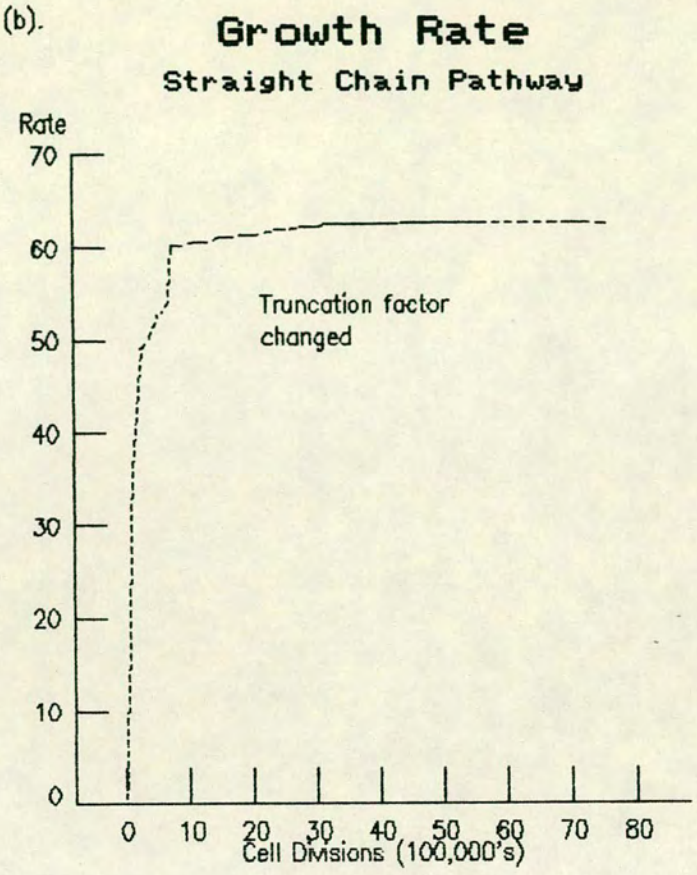
d) Variation in growth rate

e) Variation in size and complexity of genomes

(a).



(b).



(c).

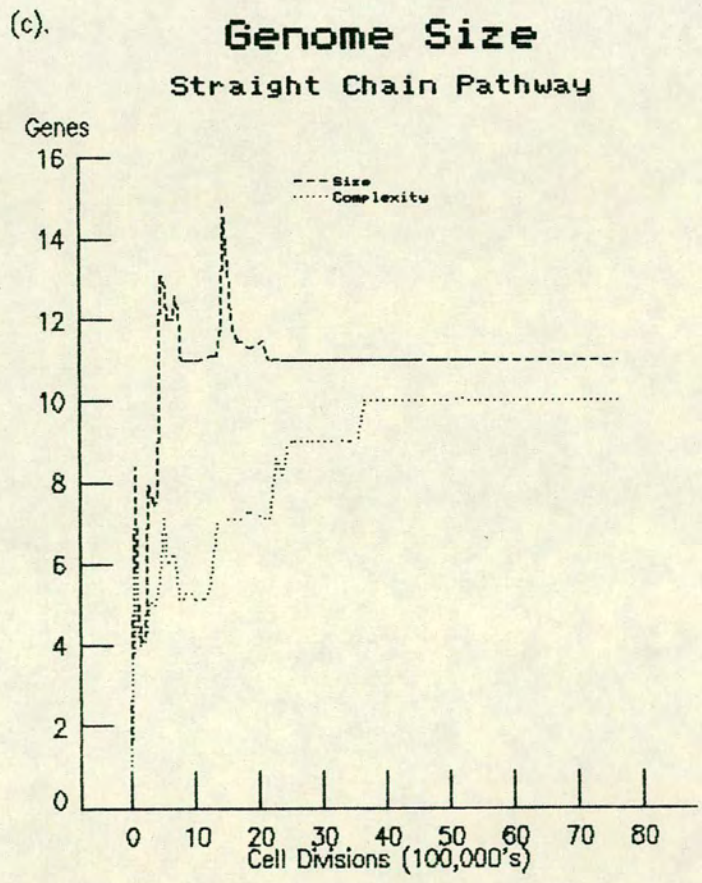
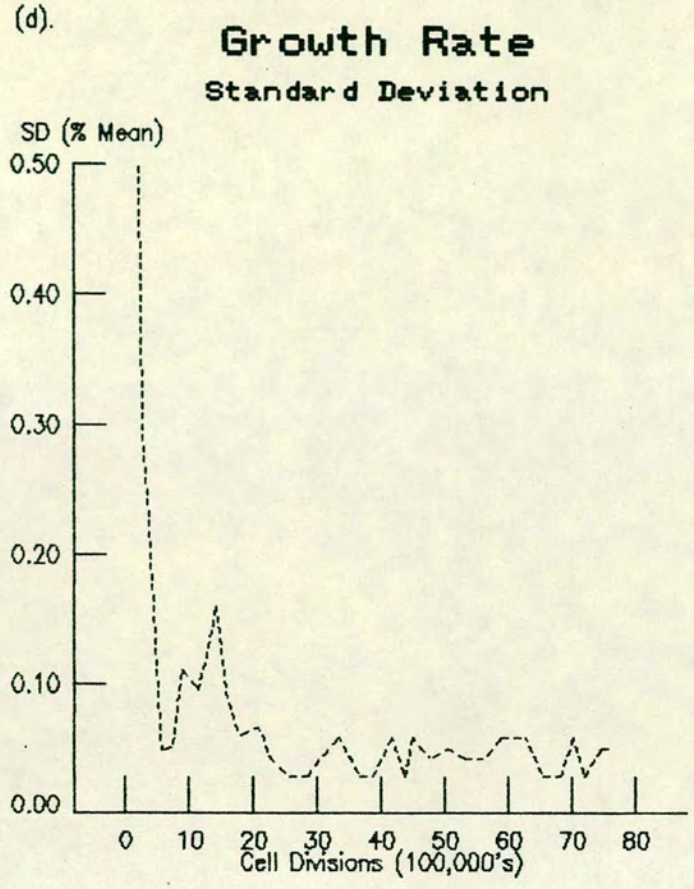


Figure 6.7.2

(d).



(e).

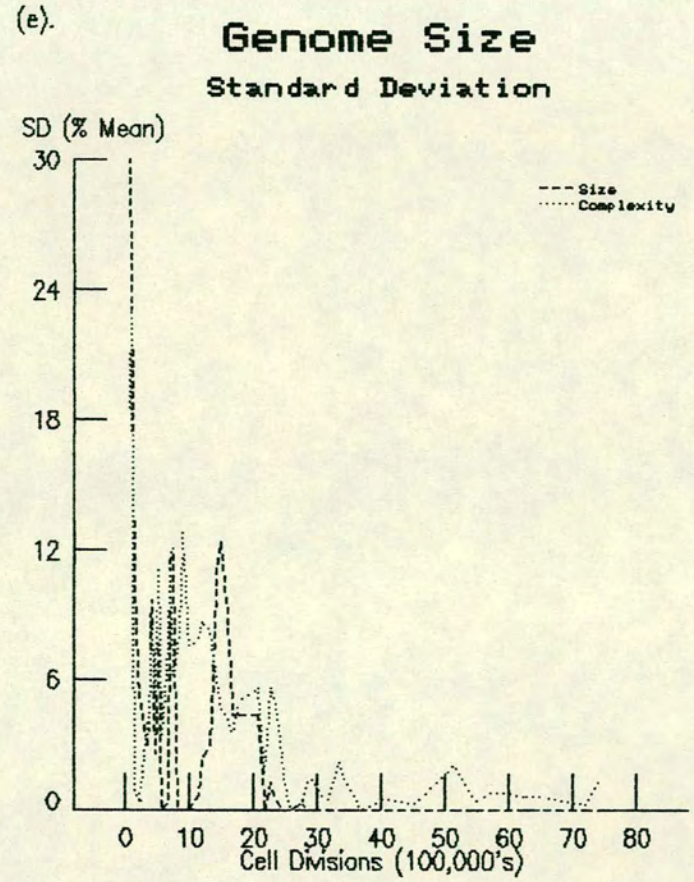


Figure 6.7.2

Experiment 6

close to the maximum 50.0 obtainable with two axes contributing when there is no "overall fit addition" being used. The binding in the third axis yields an interaction energy close to zero, hence the absence of any contribution from the addition, which the reader will recall is a scaled product of the three axis interaction energies. The very high activity form appears suddenly, apparently as the result of a mutation in a proto-enzyme with activity very close to 50. The result is a "hopeful monster" (to use Goldschmidt's, 1940, term) with a growth rate nearly ten per cent higher than that of its cousins. The general course of the experiment is shown in figures 6.7.2, 6.7.3, and 6.7.4.

At first sight, it seems that the evolution of only a single monofunctional species is due to the absence of appropriate mutations. This is illustrated by table 6.6.6 which uses an arbitrary cell sampled at the end of the experiment. The cell cannot be improved by any single duplication or deletion of any gene. The usual procedure of increasing all gene dosages five-fold (table 6.6.6a) to investigate the proximity of the relative gene dosages to optimum allows some fine-tuning of protein allocation though the increase in growth rate is only about 0.2%. What happens when one of the genes mutates to a maximum activity monofunctional one? Replacing the proto-enzyme with the smallest x-axis (activity for steps two and four) with a monofunctional proto-enzyme with maximum activity for the first metabolic step actually reduces growth rate (table 6.6.6b) by 4.5%, though this not a plausible substitution as all three proto-enzyme axes would require to be changed considerably. However, re-adjustment of the relative protein allocation in this case by one gene duplication and one deletion results in a cell with a growth rate higher than the original sampled cell. When, in table 6.6.6c, the more plausible

=> *344*

Experiment 6;

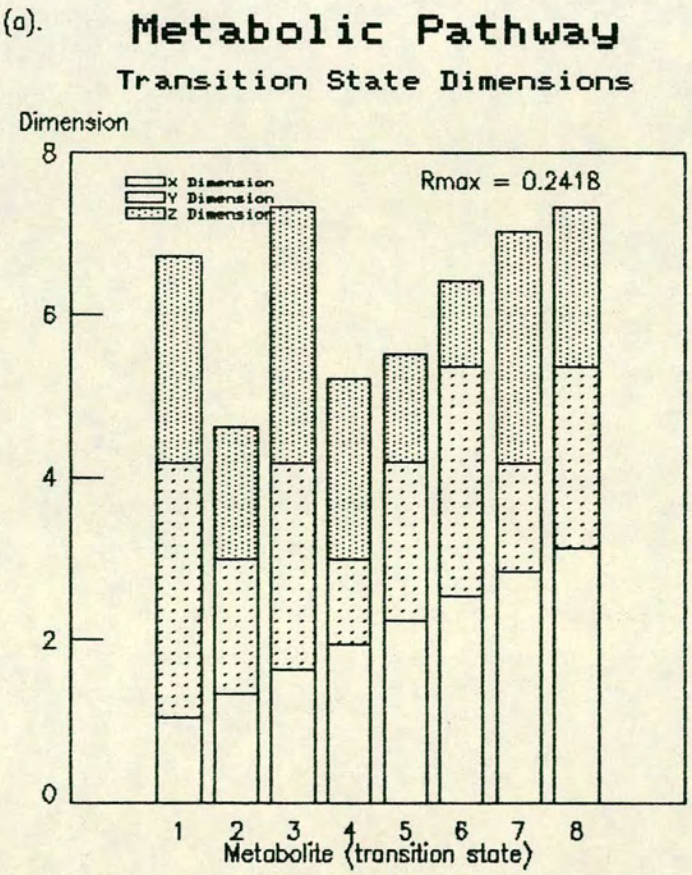
Figure 6.7.3

a) Metabolic pathway – eight reactions with no identical or similar axes dimensions

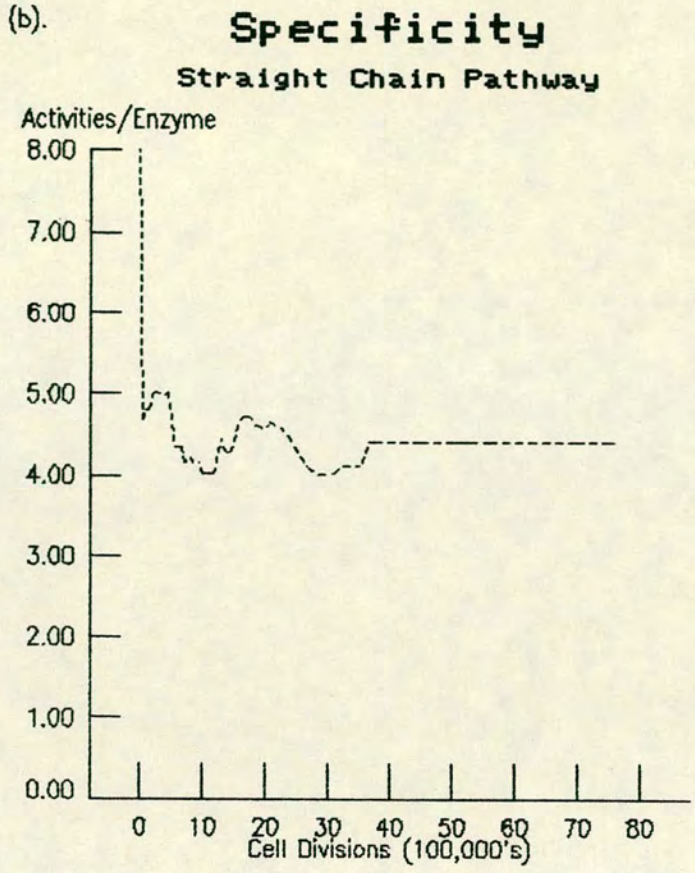
b) Mean number of activities per proto–enzyme

c) Net activities of individual cells sampled after the number of cell divisions stated

(a).



(b).



(c).

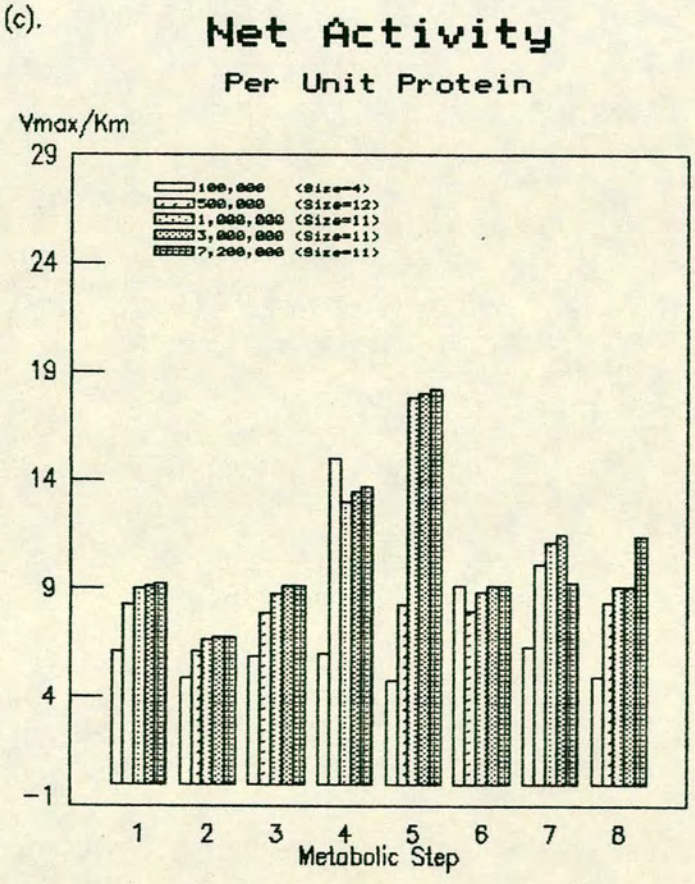


Figure 6.7.3

Experiment 6

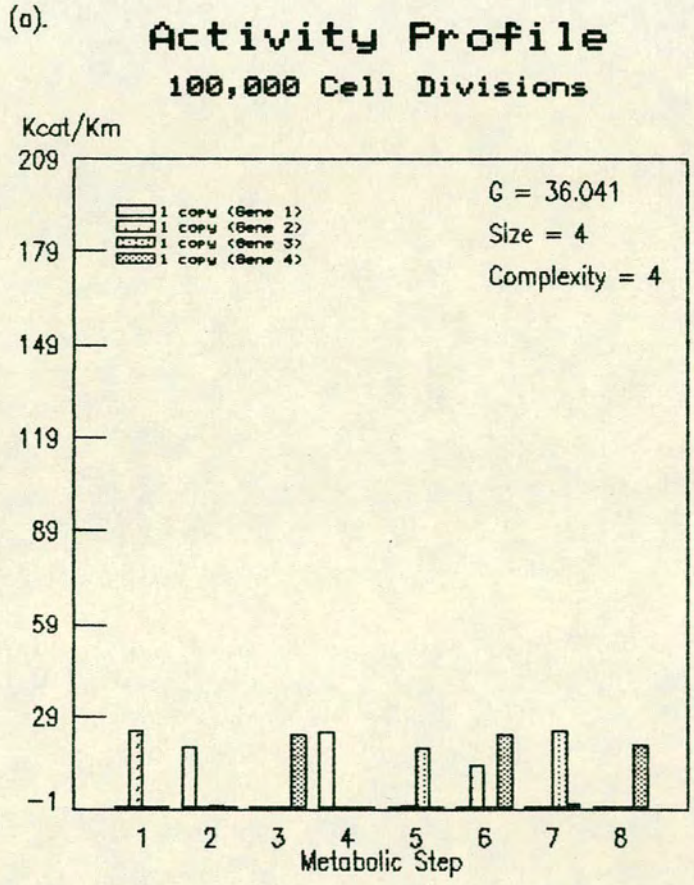
Experiment 6;

Figure 6.7.4

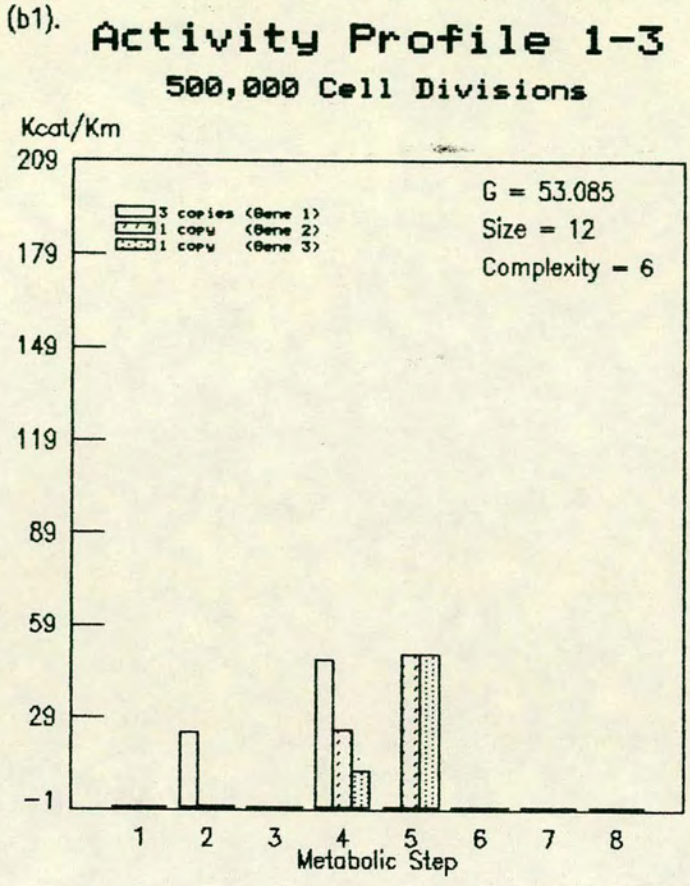
Activity profiles of cells sampled after

a) 100,000 cell divisions
b) 500,000 cell divisions
c) 1,000,000 cell divisions
d) 2,000,000 cell divisions
e) 3,000,000 cell divisions
f) 4,500,000 cell divisions
g) 7,200,000 cell divisions

(a).



(b1).



(b2).

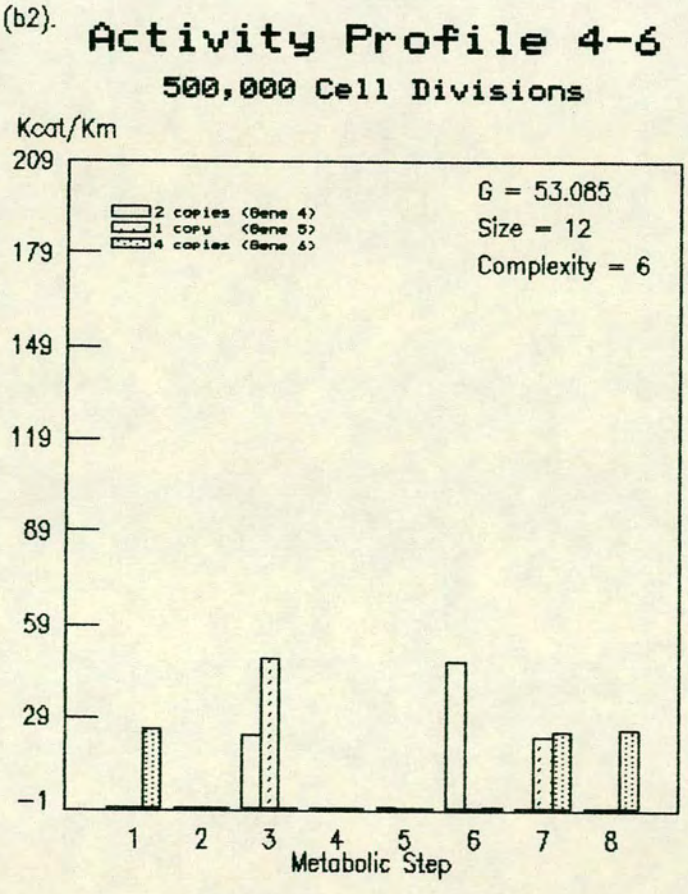
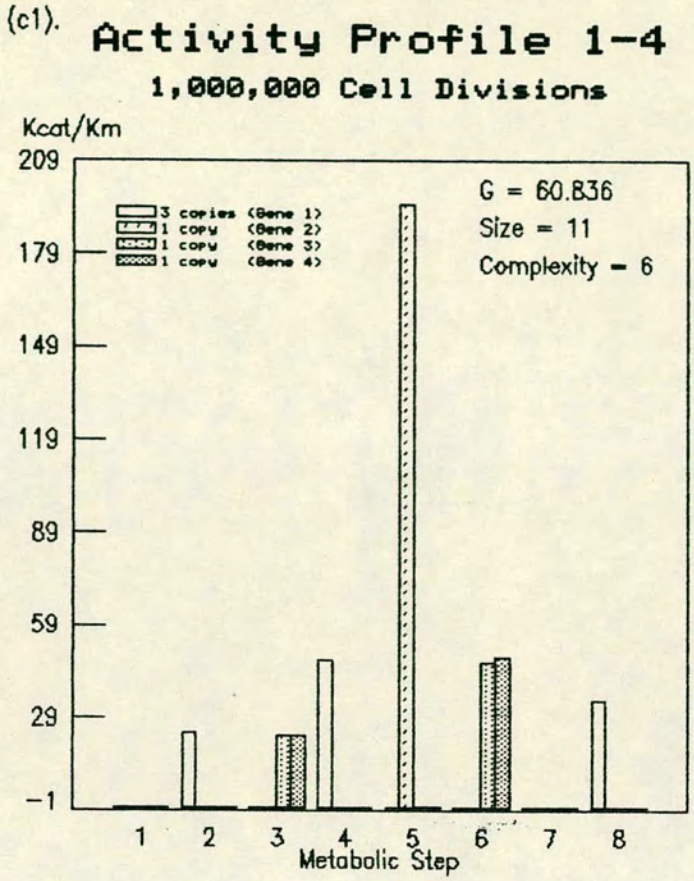


Figure 6.7.4 *Experiment 6* 335

(c1).



(c2).

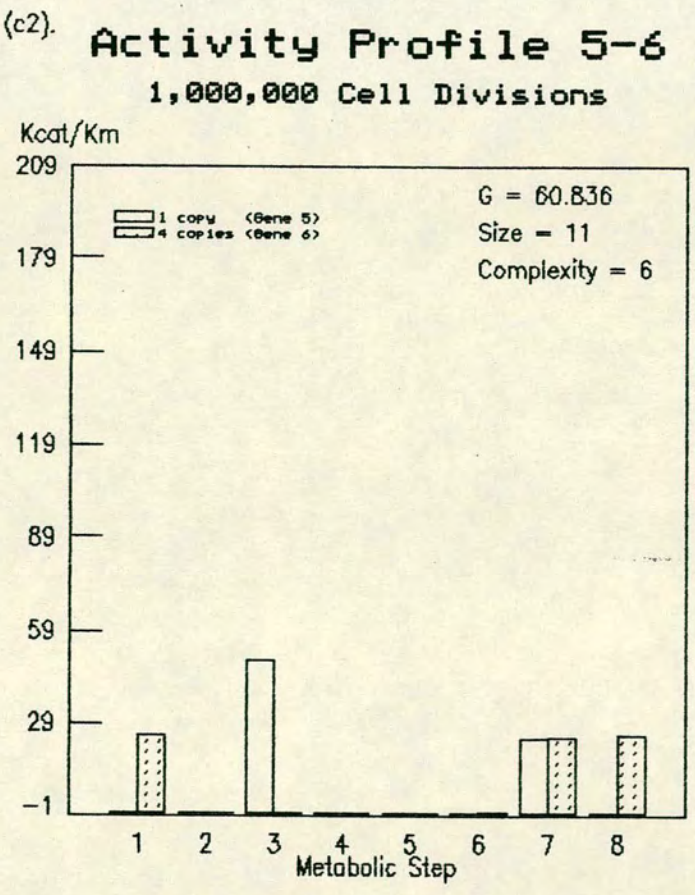
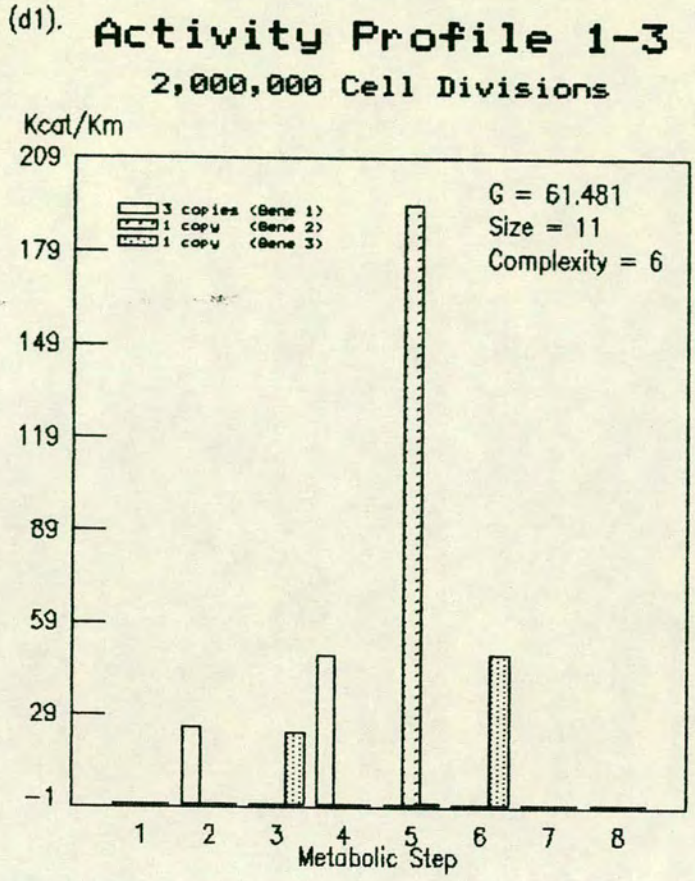


Figure 6.7.4 *Experiment 6* 336

(d1).



(d2).

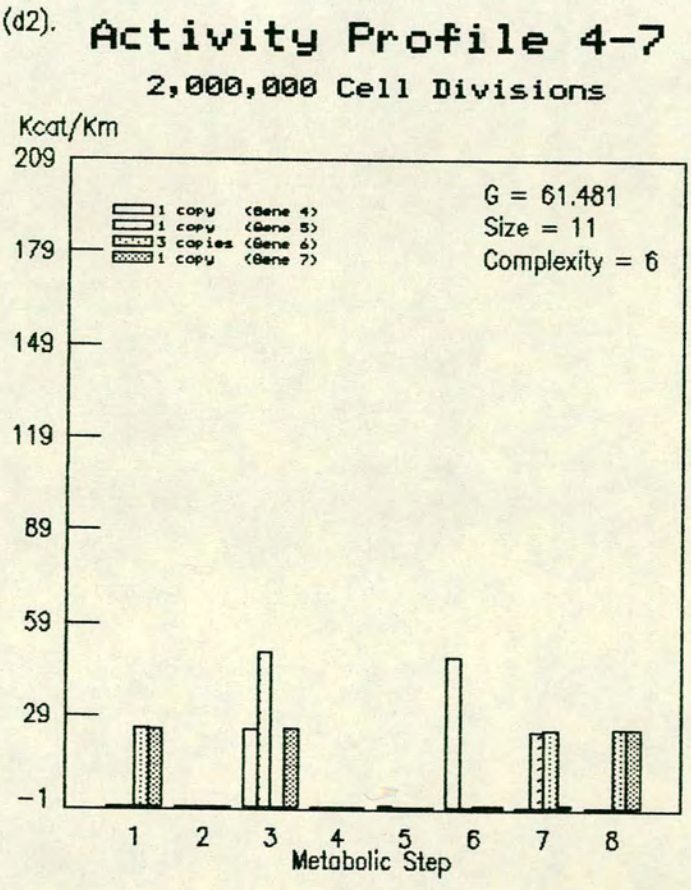
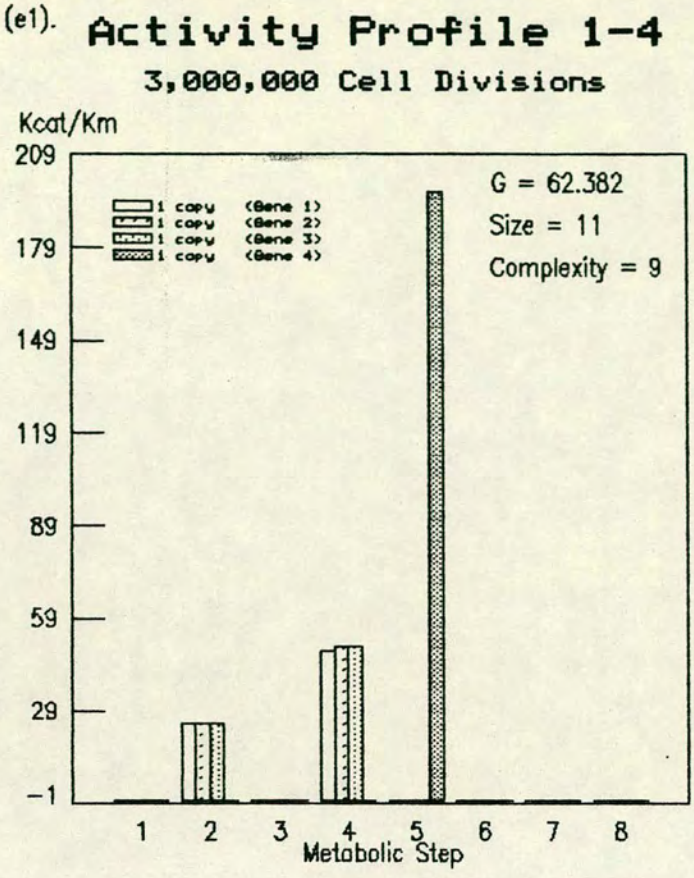


Figure 6.7.4

(e1).



(e2).

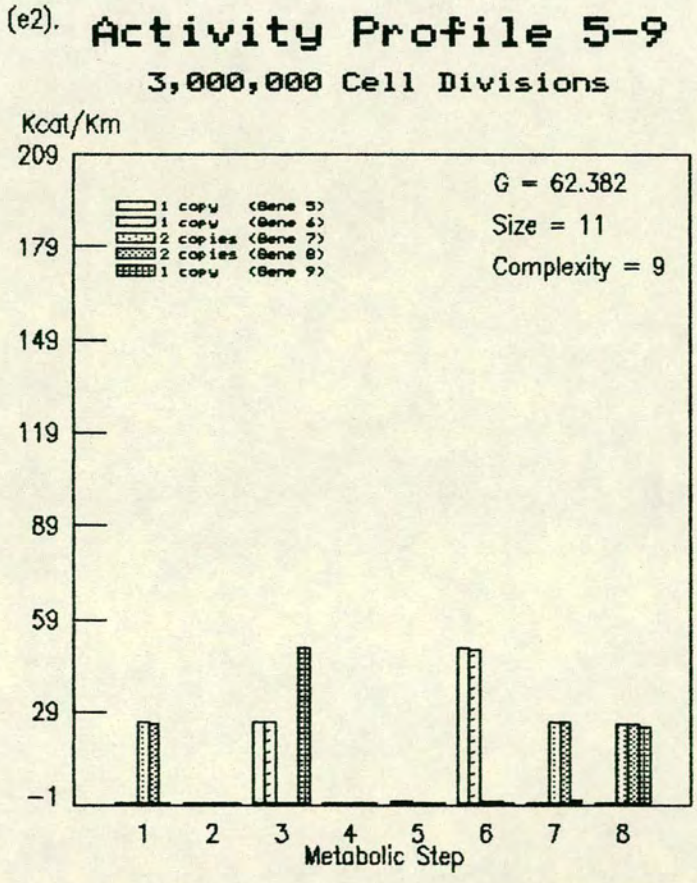
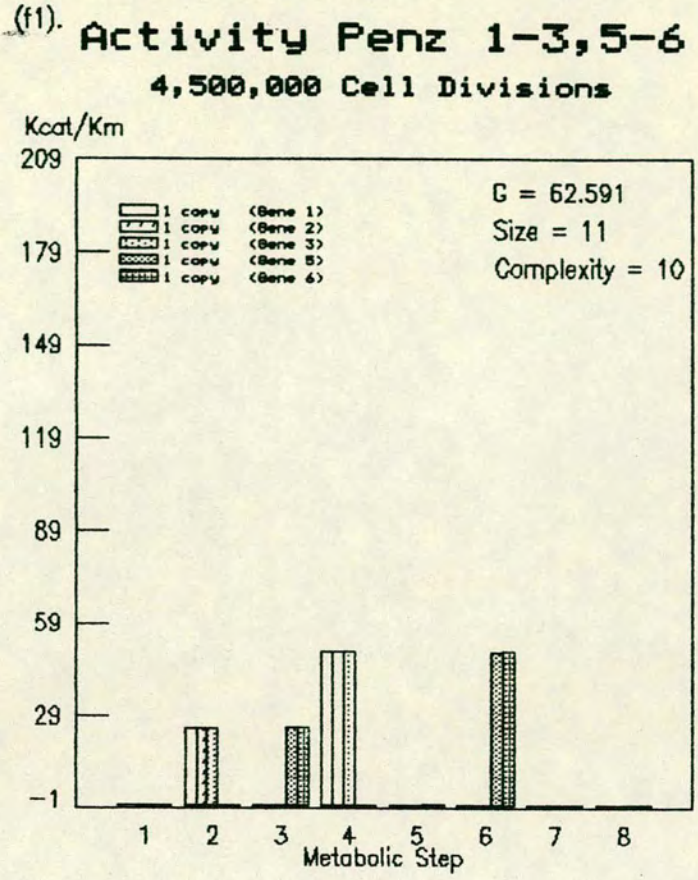


Figure 6.7.4 *Experiment 6* 338

(f1).



(f2).

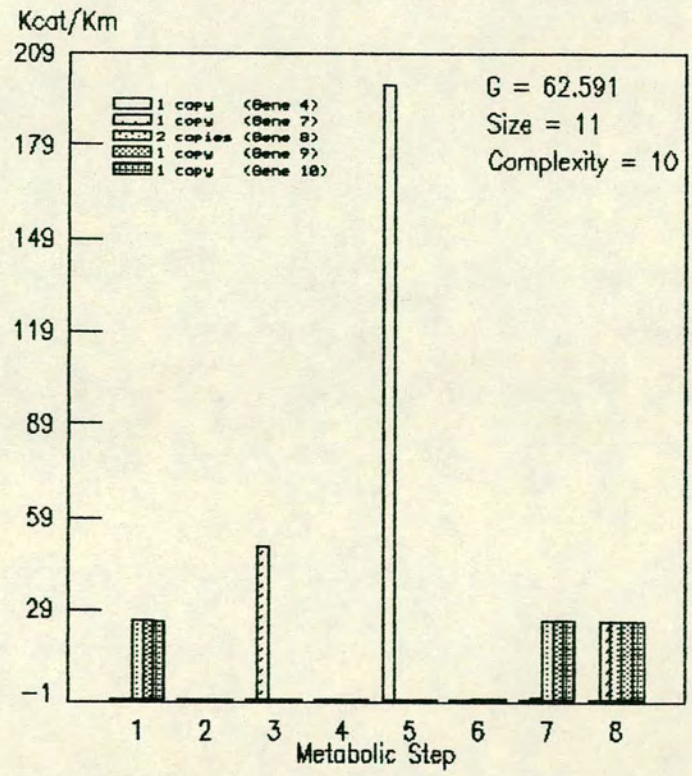
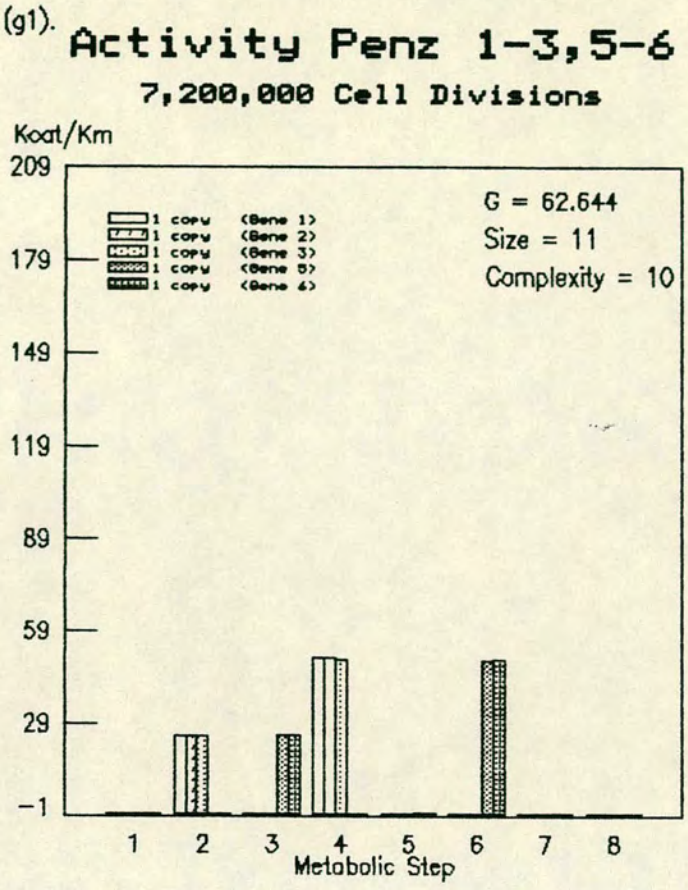


Figure 6.7.4 *Experiment 6* 339

(g1).



(g2).

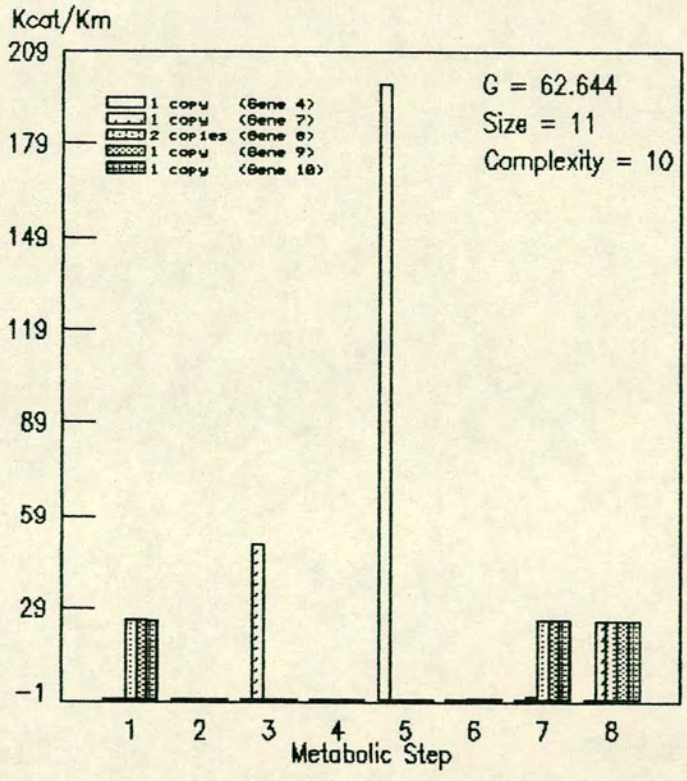


Figure 6.7.4 *Experiment 6* 340

Table 6.6.6

(a) CELL SAMPLED AFTER 7,200,000 CELL DIVISIONS OF EXPERIMENT 6 WITH ALL GENE DOSAGES INCREASED FIVE-FOLD

Genome Size = 55
Genome Complexity = 10
Growth Rate = 62.64426

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	0.00	24.97	0.00	50.02	0.00	0.00	0.00	0.00
2	5	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	5	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	5	0.00	0.00	0.00	0.00	198.83	0.00	0.00	0.00
5	5	0.00	0.00	25.30	0.03	0.43	49.46	0.00	0.00
6	5	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	5	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	10	25.38	0.00	0.00	0.03	0.01	0.43	25.38	24.95
9	5	25.39	0.00	0.00	0.03	0.01	0.42	25.38	24.95
10	5	24.90	0.00	0.00	0.03	0.01	0.40	25.38	24.95
V_{max}/K_M		9.19	6.82	9.13	13.64	18.16	9.16	9.30	11.36

REPEATED GENE DUPLICATIONS/DELETIONS IN CELL (SELECTING FASTEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
54	10	Delete	4	62.72820	5 6;	
55	10	Duplicate	6	62.75345	5;	8 9 10
54	10	Delete	7	62.75390		
54	10	None	0	62.75390		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 54
Genome Complexity = 10
Growth Rate = 62.75390

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	5	0.00	24.97	0.00	50.02	0.00	0.00	0.00	0.00
2	5	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	5	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	4	0.00	0.00	0.00	0.00	198.83	0.00	0.00	0.00
5	5	0.00	0.00	25.30	0.03	0.43	49.46	0.00	0.00
6	6	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	4	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	10	25.38	0.00	0.00	0.03	0.01	0.43	25.38	24.95
9	5	25.39	0.00	0.00	0.03	0.01	0.42	25.38	24.95
10	5	24.90	0.00	0.00	0.03	0.01	0.40	25.38	24.95
V_{max}/K_M		9.36	6.95	8.84	13.90	14.82	10.25	9.46	11.10

(b)

CELL SAMPLED AFTER 7,200,000 CELL DIVISIONS OF EXPERIMENT 6 WITH GENE 1 REPLACED BY GENE CODING FOR A MONOFUNCTIONAL PROTO-ENZYME WITH MAXIMUM ACTIVITY FOR THE FIRST METABOLIC STEP

Genome Size = 11
Genome Complexity = 10
Growth Rate = 59.83262

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	200.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	1	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.84	0.00	0.00	0.00
5	1	0.00	0.00	25.30	0.03	0.43	49.46	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	1	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.39	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.39	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.39	24.95
V_{max}/K_M		27.37	4.55	9.13	9.09	18.16	9.16	9.30	11.36

REPEATED GENE DUPLICATIONS/DELETIONS IN CELL (SELECTING FASTEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
12	10	Duplicate	2	63.16555	3;	1 8 9 10
11	9	Delete	10	63.55040	5 6;	8 9
11	9	None	0	63.55040		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 11
Genome Complexity = 10
Growth Rate = 63.55040

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	200.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	2	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.84	0.00	0.00	0.00
5	1	0.00	0.00	25.30	0.03	0.43	49.46	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	1	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.39	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.39	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.39	24.95
V_{max}/K_M		25.11	6.83	9.13	13.65	18.16	9.12	7.00	9.09

(c)

CELL SAMPLED AFTER 7,200,000 CELL DIVISIONS OF EXPERIMENT 6 WITH GENE 5 REPLACED BY A GENE CODING FOR A MONOFUNCTIONAL PROTO-ENZYME WITH MAXIMUM ACTIVITY FOR THE SIXTH METABOLIC STEP

Genome Size = 11
Genome Complexity = 10
Growth Rate = 64.95293

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	24.97	0.00	50.02	0.00	0.00	0.00	0.00
2	1	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.84	0.00	0.00	0.00
5	1	0.00	0.00	0.00	0.00	0.00	200.00	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	1	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.39	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.39	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.39	24.95
V_{max}/K_M		9.19	6.82	6.83	13.64	18.12	22.84	9.30	11.36

REPEATED GENE DUPLICATIONS/DELETIONS IN CELL (SELECTING FASTEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
12	10	Duplicate	7	65.87407		
11	9	Delete	6	68.58566	1 2 3;	
11	9	None	0	68.58566		

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 11
Genome Complexity = 10
Growth Rate = 68.58566

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	24.97	0.00	50.02	0.00	0.00	0.00	0.00
2	1	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.84	0.00	0.00	0.00
5	1	0.00	0.00	0.00	0.00	0.00	200.00	0.00	0.00
6	0								
7	2	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.39	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.39	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.39	24.95
V_{max}/K_M		9.19	6.82	9.06	13.64	18.08	18.33	9.38	13.64

replacement of gene 5's product, which has close to maximum binding for two of the three axes of the reaction intermediate for step six, with a monofunctional proto-enzyme with maximum activity for that step, gives an immediate increase in growth rate of about 3.7%. Further improvement is obtained after tuning the relative gene dosages (and hence protein allocation).

In the cell shown in table 6.6.6 there are six genes (1, 2, 3, 5, 6, 7) coding for bifunctional protoenzymes which have almost maximum binding in two of the three dimensions for one of the reaction intermediates. This is a difference from previous experiments, where trifunctionality, due to only one axis being maximally bound, was the rule. The table below shows the six protoenzymes, and also the dimensions of the corresponding maximally binding monofunctional protoenzyme:

Table 6.6.7

Enzyme/Gene	*Axis Dimensions*		
	X	*Y*	*Z*
Monofunctional step 4	2.18180	1.28180	2.48180
1	2.18069	1.87953	2.47902
2	2.18086	1.88216	2.47902
3	2.18772	1.87953	2.48087
Monofunctional step 6	2.78180	3.08180	1.28180
5	2.77846	3.08692	3.37964
6	2.78483	3.08617	3.38449
Monofunctional step 3	1.88180	2.78180	3.38180
7	3.37804	2.78517	3.38317

These genes are the obvious candidates for mutational movement to monofunctionality, as only one axis needs to be altered. In the case of genes 1, 2 and 3 this does mean that a single mutation can produce a (very close to) maximum activity monofunctional catalyst. This is not so for genes 5, 6, and

7, however, for the relative change in the size of the axis to be altered exceeds the largest allowed under the mutation scheme (two thirds). The conservatism of the mutational process in the model is therefore a constraint on the evolutionary potential of the population.

Table 6.6.8 below shows this for gene 5. The z-axis of the protoenzyme coded for has been reduced as far as is allowed under the present mutation scheme, bringing it as close as possible (in a single step) to the dimensions of the maximum activity monofunctional protoenzyme. This does result in the exclusion of the reaction intermediate for the third metabolic step, and leaves step six as the only significant catalytic activity of the protoenzyme, but the increase in activity for step six, from 49.46 to 49.48, provides no compensation for the activity lost by excluding the reaction species for step three, and growth rate is adversely affected. Even after gene 7 is duplicated (increasing the net activity for step three) the growth rate is still considerably lower than the original cell. The evolution of monofunctionality in the gene lineages of genes 5, 6, and 7 seems therefore excluded. This is not so for the lineages of genes 1, 2, and 3, however, and presumably suitable mutations have simply not occurred.

Table 6.6.8

CELL SAMPLED AFTER 7,200,000 CELL DIVISIONS OF EXPERIMENT 6 WITH THE MAXIMUM POSSIBLE SINGLE STEP REDUCTION IN THE Z AXIS OF THE PROTOENZYME CODED BY GENE 5 INTRODUCED BY MUTATION

Genome Size = 11
Genome Complexity = 10
Growth Rate = 59.90020

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	24.97	0.00	50.02	0.00	0.00	0.00	0.00
2	1	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.84	0.00	0.00	0.00
5	1	0.00	0.17	0.00	0.00	0.45	49.48	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	1	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.39	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.39	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.39	24.95
V_{max}/K_M		9.19	6.83	6.83	13.64	18.16	9.16	9.30	11.36

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
12	10	Duplicate	7	60.25107	6	
12	10	No further favourable events				

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 12
Genome Complexity = 10
Growth Rate = 60.25107

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	24.97	0.00	50.02	0.00	0.00	0.00	0.00
2	1	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.84	0.00	0.00	0.00
5	1	0.00	0.17	0.00	0.00	0.45	49.48	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	2	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.39	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.39	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.39	24.95
V_{max}/K_M		8.42	6.27	10.41	12.50	16.65	8.40	8.59	12.51

It therefore seems that the monofunctional protoenzyme is inaccessible to the population for at gene lineages starting with genes 5, 6, or 7. However, they are accessible to genes 1, 2, and 3. Is the failure of these to evolve due to the absence of suitable mutations? Table 6.6.9 addresses this question:

Table 6.6.9

CELL SAMPLED AFTER 7,200,000 CELL DIVISIONS OF EXPERIMENT 6 WITH GENE 1 REPLACED BY A GENE CODING FOR A MAXIMUM ACTIVITY PROTOENYME WITH ACTIVITY FOR THE FOURTH METABOLIC STEP

Genome Size = 11
Genome Complexity = 10
Growth Rate = 59.89417

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	0.00	0.00	199.09	0.00	0.00	0.00	0.00
2	1	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.83	0.00	0.00	0.00
5	1	0.00	0.00	25.30	0.03	0.43	49.46	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	1	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.38	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.38	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.38	24.95
V_{max}/K_M		9.19	4.55	9.13	27.19	18.16	9.16	9.30	11.36

REPEATED DUPLICATIONS/DELETIONS IN CELL (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications	Deletions
12	10	Duplicate	2	60.61147	3;	1
11	9	Delete	*1*	62.66237	6 8 9;	
11	9	No further favourable events				

The new gene 1, coding for the monofunctional protoenzyme, is deleted!

FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 11
Genome Complexity = 9
Growth Rate = 62.66237

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	0								
2	2	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.83	0.00	0.00	0.00
5	1	0.00	0.00	25.30	0.03	0.43	49.46	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	1	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.38	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.38	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.38	24.95
V_{max}/K_M		9.19	6.83	9.13	13.65	18.16	9.16	9.30	11.36

The problem revealed by the table above is that mutation of genes 1, 2, or 3 to yield a monofunctional protoenzyme has a dramatic effect on the activity for the second metabolic step, which is entirely carried by the products of these three genes. So much so that although the activity for step four is easily the highest in the cell after the mutation of gene 1 the only favourable gene copy changes are duplication of genes 2 or 3, whose major contribution is to that same activity (but which also are the only remaining contributors to the activity for step two). Once a duplication of gene 2 or 3 has occurred, the cell can restore itself approximately to its original state by deleting the gene for the monofunctional protoenzyme, and this turns out to be the cells best (though not sole) option. It therefore seems that the present population is unlikely to evolve any additional monofunctional protoenzymes.

In fact, there is a route allowing such evolution, and this is shown in table 6.6.10 below. This depends on a mutation in one of genes 1, 2, or 3 which excludes the reaction intermediate for the fourth metabolic step, and creates a monofunctional protoenzyme which binds the transition state of the second metabolic step in two dimensions. As table 6.6.10A shows, this is slightly favourable (growth rate increases by about 0.02%) and is allowed by the mutation scheme. The final axis of the protoenzyme can then be altered (table 6.6.10B) to give a maximum activity protoenzyme, with a considerable gain in growth rate (again, this is a possible one-step mutation). In this cell, either of the remaining two bifunctional protoenzymes for steps two and four can be altered to give a maximum activity protoenyzme for step four (table 6.6.10C), and the other (unaltered) protoenzyme eliminated (table 6.6.10D). At this point, however, no further mutational changes in the direction of monofunctionality for the remaining multifunctional protoenzymes is possible because the mutational limit does not allow any of the protoenzymes to approach the maximum activity form closely enough to compensate for the losses in activity which must be incurred.

That such a sequence did not occur in the simulation is presumably due to the non-occurrence or non-survival of the initial mutation.

Genes 8, 9, and 10 have no favourable mutations in the direction of monofunctionality. Once again, the cell is trapped in a sub-optimal position by its multifunctionality. In this case, however, the mutational model has also made itself felt, preventing three of the genes present (5, 6, and 7) from

"leaping" to maximum monofunctionality (the only route which does not involve a lower growth-rate intermediate).

Table 6.6.10

(A) CELL SAMPLED AFTER 7,200,000 CELL DIVISIONS OF EXPERIMENT 6 WITH THE X-AXIS OF PROTOENZYME 1 ALTERED TO INCREASE ACTIVITY FOR THE SECOND METABOLIC STEP AT THE EXPENSE OF THE FOURTH

Genome Size = 11
Genome Complexity = 10
Growth Rate = 62.65480

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	50.08	0.00	0.00	0.00	0.00	0.00	0.00
2	1	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.83	0.00	0.00	0.00
5	1	0.00	0.00	25.30	0.03	0.43	49.46	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	1	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.38	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.38	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.38	24.95
V_{max}/K_M		9.19	9.10	9.13	9.09	18.16	9.16	9.30	11.36

(B) THE ABOVE CELL, WITH THE Z-AXIS OF PROTOENZYME 1 ALSO ALTERED TO GIVE MAXIMUM CATALYTIC ACTIVITY FOR THE SECOND METABOLIC STEP

Genome Size = 11
Genome Complexity = 10
Growth Rate = 68.28316

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	199.48	0.00	0.00	0.00	0.00	0.00	0.00
2	1	0.00	25.05	0.00	50.15	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.83	0.00	0.00	0.00
5	1	0.00	0.00	25.30	0.03	0.43	49.46	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	1	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.38	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.38	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.38	24.95
V_{max}/K_M		9.19	22.68	9.13	9.09	18.16	9.16	9.30	11.36

(C) THE ABOVE CELL, WITH THE Y-AXIS OF THE GENE 2 PRODUCT ALTERED TO GIVE A MAXIMUM-ACTIVITY ENZYME FOR THE FOURTH METABOLIC STEP

Genome Size = 11
Genome Complexity = 10
Growth Rate = 74.48805

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	199.48	0.00	0.00	0.00	0.00	0.00	0.00
2	1	0.00	0.00	0.00	199.91	0.00	0.00	0.00	0.00
3	1	0.00	24.97	0.00	49.70	0.00	0.00	0.00	0.00
4	1	0.00	0.00	0.00	0.00	198.83	0.00	0.00	0.00
5	1	0.00	0.00	25.30	0.03	0.43	49.46	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	1	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.38	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.38	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.38	24.95
V_{max}/K_M		9.19	20.41	9.13	22.71	18.16	9.16	9.30	11.36

REPEATED DUPLICATIONS/DELETIONS IN CELL (C) (SELECTING HIGHEST RATE)

Genome Size	Complexity	Event	Gene	Growth Rate	Other Favourable Duplications Deletions
10	9	Delete	3	79.90470	5 6 8 9 10;
10	9	No further favourable events			

(D) FINAL CELL WITH NO FAVOURABLE SINGLE GENE DUPLICATIONS/DELETIONS

Genome Size = 10
Genome Complexity = 9
Growth Rate = 79.90470

Gene	Copies	k_{cat}/K_M (Metabolic Step) 1	2	3	4	5	6	7	8
1	1	0.00	199.48	0.00	0.00	0.00	0.00	0.00	0.00
2	1	0.00	0.00	0.00	199.91	0.00	0.00	0.00	0.00
3	0								
4	1	0.00	0.00	0.00	0.00	198.83	0.00	0.00	0.00
5	1	0.00	0.00	25.30	0.03	0.43	49.46	0.00	0.00
6	1	0.00	0.00	25.28	0.03	0.40	49.62	0.00	0.00
7	1	0.00	0.00	49.82	0.01	0.03	0.00	0.80	25.13
8	2	25.38	0.00	0.00	0.03	0.01	0.43	25.38	24.95
9	1	25.39	0.00	0.00	0.03	0.01	0.42	25.38	24.95
10	1	24.90	0.00	0.00	0.03	0.01	0.40	25.38	24.95
V_{max}/K_M		10.11	19.95	10.04	20.01	19.97	10.07	10.23	12.49

A Branched Chain of Monomolecular Reactions

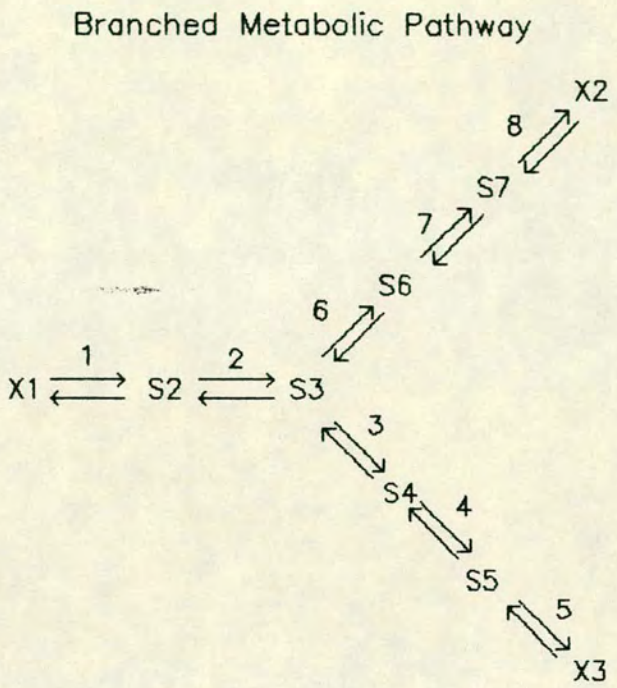
At the end of chapter 3, some attention was given to a branched system of unimolecular reactions. Such a system was shown in figure 3.2 (page 85). The inclusion of such a system in the present work allows investigation of the evolutionary behaviour of simple systems other than the straight chain without incurring additional heavy computational overheads (the system has linear kinetics).

To represent an organism, the output of the system is required to express the rate of growth, and with two outputs the simplest scheme is to regard one branch as a flux to "waste". This allows selection for discrimination against some particular set of subtrates to be applied. In this section I will briefly discuss the results of several simulations using a branched chain scenario though data on the progress of particular simulations will be presented for only one of them.

Figure 6.8.1a shows the pathway used in the experiments described here. As the the figure shows, there are three branches to the system. That comprising the two steps leading from X_1 to S_3 represents the "common branch" through which all the molecules transformed by the system must pass. S_3 to X_2 (three steps in all) represents the "waste branch", and the three steps from S_3 to X_3 represents the "branch to growth". S_3 is a common substrate for the two outputs, therefore, with the waste branch transforming the molecule into some unusable product.

Figure 6.8.1

(a).



(b).

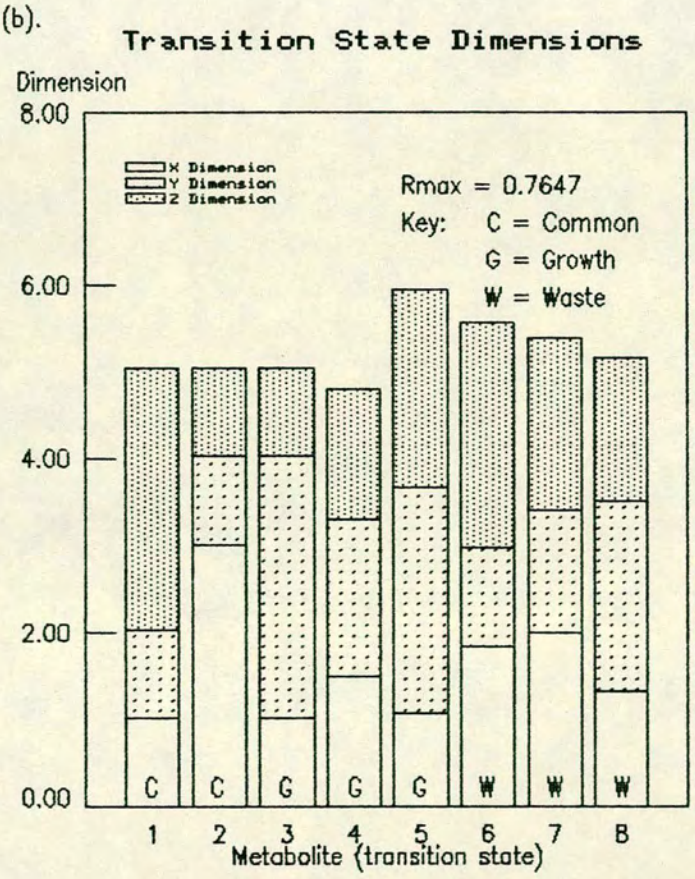
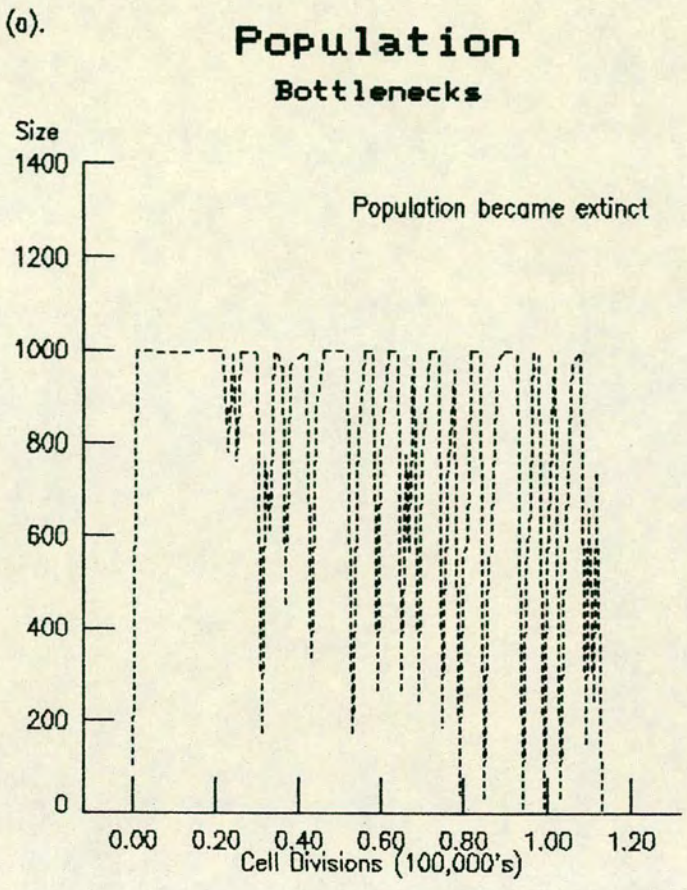


Figure 6.8.1

In all the experiments to be referred to, the reaction intermediates and Lennard-Jones constants used in the introductory and subsequent experiments are reused, with steps 3 to 5 assigned to the branch to growth and 6, 7, and 8 to the waste branch (figure 6.8.1b). The first two steps form the common branch.

For the particular set of equilibrium constants and external concentrations chosen, the growth rates for the branched scenario are higher than the straight chain case. The thermodynamic "pressure" for the two competing output branches was set to be the same, so that their relative fluxes would reflect the relative levels of activity of their respective catalysts. A cell with eight monofunctional maximum-activity protoenzymes (one for each reaction) would have equal fluxes to waste and to growth, and a growth rate of 66.96.

The results of a short experiment (F1) are shown in figure 6.8.2, which represents a single run of the simulation program that was terminated after some 113,000 cell divisions because the population became extinct (figure 6.8.2a). The truncation factor in the experiment was 1.00, and there was no appreciable variation in growth rate at the end of the experiment (figure 6.8.2e). The size of the population fluctuated considerably throughout the experiment because of the fatalities due to truncation selection. Despite the brevity of the experiment, the final growth rate, which exceeded 156 (figure 6.8.2b), was the second highest achieved in the dozen or so experiments performed with this scenario (though as we shall see below they are not all directly comparable).



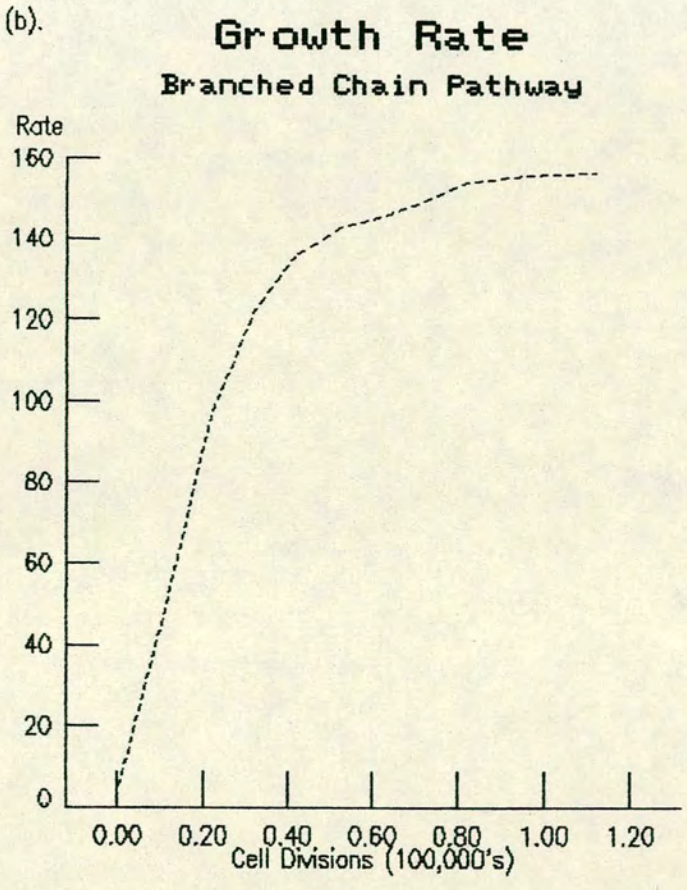
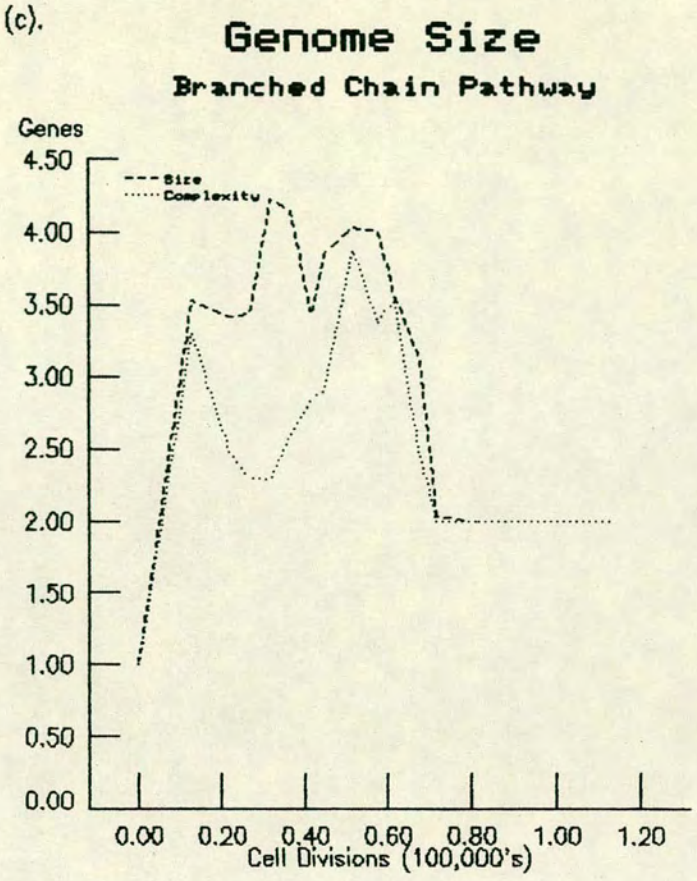


Figure 6.8.2

Experiment F1

(c).



(d).

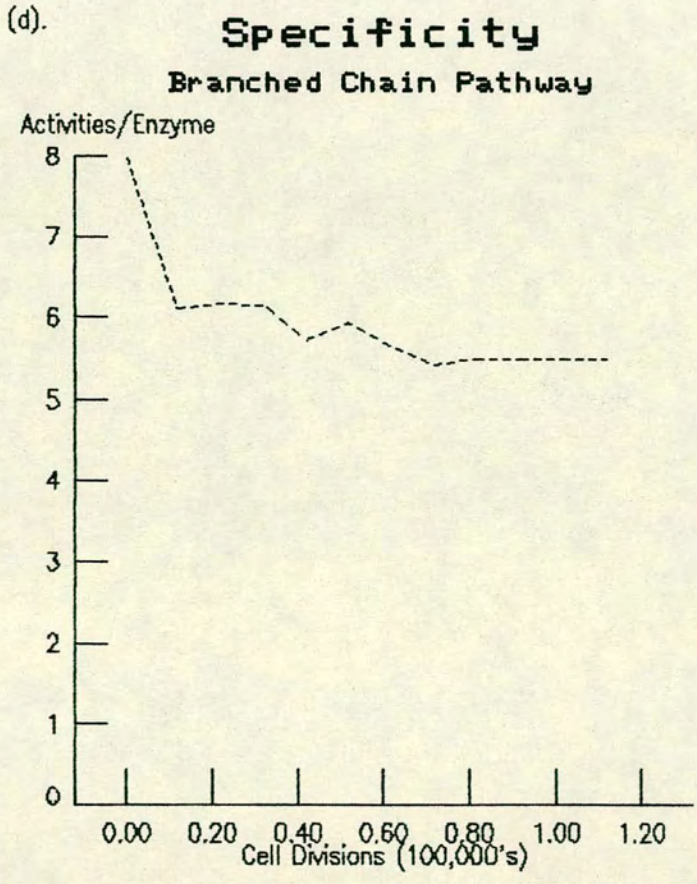
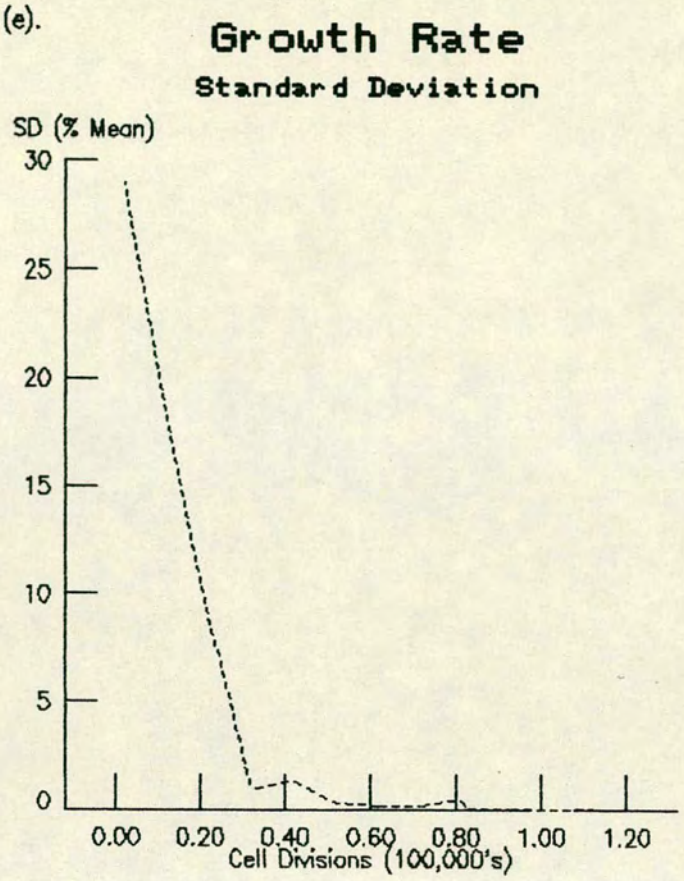


Figure 6.8.2 *Experiment F1* 357

(e).



(f).

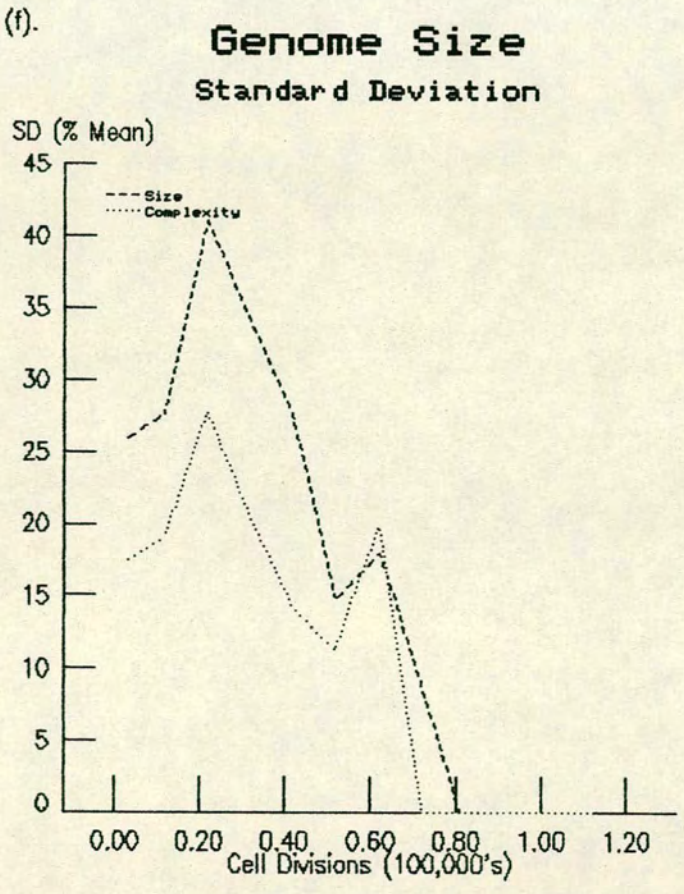


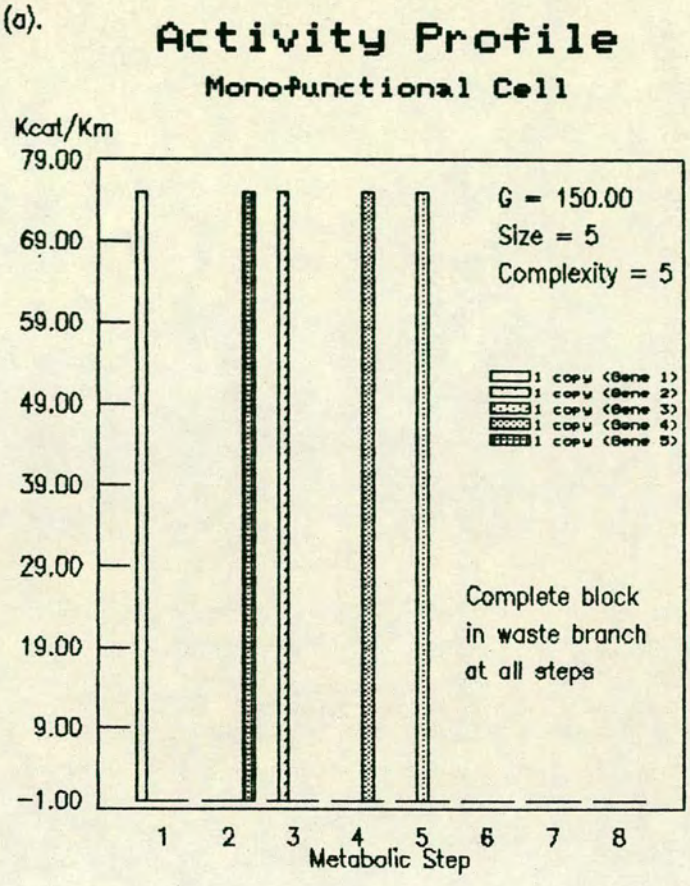
Figure 6.8.2

In most of these experiments, the populations rapidly converged on a genome size and complexity of two, as in the present case (figure 6.8.2c). No variation in either the size or complexity of genomes is present at the end of the experiment (figure 6.8.2f). The five activities contributing to growth can obviously be maintained by two protoenzymes if the previously encountered pattern of three activities per catalyst (corresponding to each of the three molecular axes) is followed. The mean number of activities per protoenzyme in the present experiment is actually nearly twice this (figure 6.8.2d), but some of these, as we have seen in previous experiments, and shall see in the next figure, are low.

Figure 6.8.3a shows the activity profile for a cell with five monofunctional maximum-activity protoenzymes, corresponding to the five activities required for growth. The cell has no activity for any of the three wasteful steps, and has a growth rate of 150.00, more than double that of the monofunctional cell mentioned above with activities supporting flux through the waste path.

Part (b) of the figure shows a cell sampled towards the end of the experiment, and reveals that there are indeed three major activities for each of the two protoenzymes of the cell, and that there is a complete block in the waste branch at step 6 (between S_9 and S_6). Both protoenzymes have a major activity for step 3, with all other steps in the growth path carried primarily by one or other protoenzyme. The growth rate of this cell is 156.39, somewhat higher than the monofunctional cell.

(a).



(b).

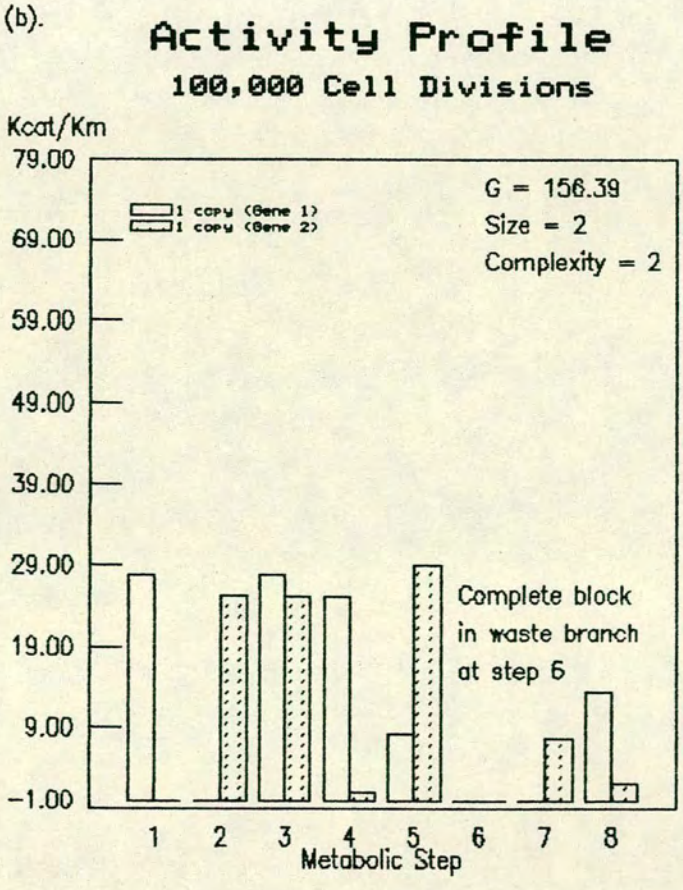


Figure 6.8.3 *Experiment F1* 360

Clearly there are many alternative scenarios that can be used to promote or disadvantage varying degrees of multifunctionality, by manipulating the overall similarity of the reaction intermediates in each branch and the relative thermodynamic pressures in driving the flux throught them. This represents potential future work. One interesting experiment that was tried, however, is to simply disallow complete blocks in the waste branch. This imposes selection to reduce the flux in the waste path to a very low, but finite, level. Admittedly, this is difficult to justify biologically, but is much more testing on the population. Seven of the experiments performed were of this kind. To restate the idea: any complete block in the system is treated as lethal, but selection to maximise the flux to X_3 will act to reduce the flux to X_2 to the minimum possible.

The surprising thing about these seven experiments was that in five of them the final growth rate of the population (three of which became extinct because of truncation selection) was between 139 and 140. In three of these five cases the cells evolved genome sizes and complexities of two, and in the other two, the final genome size with thirteen. The active site dimensions and activities from cells sampled at the end of these five experiments, which shall be numbered B1 to B5, are shown in table 6.8.1.

These experiments represent a remarkable degree of convergence. All five have found essentially the same solution to the problems imposed on them. Experiments B1 to B3 have settled to a genome size of two that has prevented them from inventing the third protoenzyme of experiments B4 and B5, which improves the relative levels of the activities a little. The protoenyzmes

evolved, and their relative doses, are otherwise the same. The throttling down of the waste branch is less effective than one might expect, with the lowest activity (for the final step from S_7 to X_2), equal to 2.0, carried by one of the two protoenzyme species (and also by the third catalyst in experiments B4 and B5).

Table 6.8.1

Experiment	*Growth Rate*	*Gene Dose*	*Protoenzyme Axes* X	Y	Z	*k_{cat}/K_M for Metabolic Step* 1	2	3	4	5	6	7	8
B1[c]	139,14	1	3,76	2,56	3,76	25,8	25,8	0,0	25,2	0,0	*5,2*	*4,5*	*0,0*
		1	3,76	3,76	3,05	0,0	25,1	25,2	1,0	28,8	*0,0*	*7,2*	*2,0*
B2	139,22	1	3,77	2,56	3,77	25,7	25,8	0,0	25,2	0,0	*5,1*	*4,5*	*0,0*
		1	3,77	3,76	3,05	0,0	25,2	25,2	1,0	28,9	*0,0*	*7,3*	*2,0*
B3[c]	139,08	1	3,76	2,56	3,76	25,7	25,7	0,0	25,1	0,0	*5,2*	*4,4*	*0,0*
		1	3,77	3,76	3,05	0,0	25,1	25,1	1,0	28,9	*0,0*	*7,2*	*2,0*
B4	139,29	6[a]	3,77	2,57	3,76	25,7	25,7	0,0	25,1	0,0	*5,3*	*4,5*	*0,0*
		6[a]	3,77	3,77	3,05	0,0	25,2	25,1	1,0	29,0	*0,0*	*7,3*	*2,0*
		1	3,77	3,76	3,76	25,0	25,0	24,9	0,3	5,0	*4,2*	*0,7*	*0,9*
B5[c]	139,64	6[b]	3,77	2,57	3,76	25,7	25,7	0,0	25,1	0,0	*5,3*	*4,5*	*0,0*
		6[b]	3,77	3,76	3,05	0,0	25,2	25,0	1,0	29,0	*0,0*	*7,2*	*2,0*
		1	3,77	3,76	3,76	25,0	25,0	24,9	0,3	5,0	*4,2*	*0,7*	*0,9*

Notes; Waste branch steps are shown in italics,

a) The genome complexity was 10 but all genes fell into the classes shown,

b) The genome complexity was 13 but all genes fell into the classes shown,

c) The population became extinct,

Clearly the solution adopted is one that is readily found. It is not, however, the only solution. Both of the other experiments evolved populations with higher growth rates than above. B6 found another two-gene solution with a final growth rate of about 142. The other, however gave rise to a population with a mean growth rate considerably higher than that of experiment F1 (in

which complete waste-branch blocks were allowed). The final cells of this experiment had the form shown in table 6.8.2 below

Table 6.8.2

Experiment	*Growth Rate*	*Gene Dose*	*Protoenzyme Axes X*	*Y*	*Z*	k_{cat}/K_M *for Metabolic Step 1*	*2*	*3*	*4*	*5*	6	7	8
B7	179,81	16	1,81	3,35	3,76	49,1	0,0	0,0	0,0	50,3	*0,0*	*0,0*	*0,0*
		17	1,81	3,76	3,76	49,6	0,0	49,1	0,0	29,3	*0,0*	*0,0*	*0,0*
		1	3,77	3,42	3,78	24,7	25,0	0,0	0,7	22,4	*3,8*	*0,8*	*3,0*
		43	3,77	3,77	2,25	0,0	27,8	27,8	24,8	0,0	*0,0*	*0,0*	*0,0*

Note; Waste branch steps are shown in italics,
The genome complexity was 9 but all genes fell into one of the four classes shown,

This is a very impressive solution to the demands on the cells. The total genome size is 77 (the mean genome size of the population was 73.4 at the end of the experiment), and only one of the genes (in single copy) has any activity for waste-branch steps. The net extractable activity for these steps is more than a thousand-fold less than for the five steps contributing to growth. The population has also made full use of the similarities in reaction intermediate dimensions for the first, third and fifth metabolic steps in the products of the first two gene classes in the table.

Chapter 7

Conclusions

Summary and Conclusions

> *Since the theory* (of natural selection) *has difficulties perhaps it should not receive such emphasis. There are alternative questions open to the researcher. The actual discovery of the patterns of nature may not necessitate a theory of their mechanism... Perhaps many researchers simply do not need natural selection to investigate their part of the world.*
>
> R. H. Brady (1979: 620).

> *My main message, therefore, is that most current versions of evolutionary theory are presented in such a vague way and include so many areas of apparent indeterminacy that too many products are possible with the proposed mechanisms and processes. By this I mean that current theories of mechanism do not constrain phylogeny very much... The relevance of neodarwinism to either phylogeny or ontogeny, in being founded on notions of genetic or ecological chance, appear to me to be simply undecipherable.*
>
> D. E. Rosen (1982: 84).

> *Phenomena over long time scales include the radiation of groups of organisms, as exemplified by the radiation of lung fish... We want to understand phenomena of this sort. Why has the radiation been as extensive as it has been? Could certain forces have caused it to be more or less extensive; what controls the rate at which the radiation unfolds; and is there any causal connection between radiation patterns in different groups? These are all important and fundamental questions, yet we cannot answer them very well. When research into population genetics was begun, it was assumed that these kinds of long term evolutionary phenomena would be explained as a result. We need to reasses the relevance of population genetics to these kinds of evolutionary issues.*
>
> J. Roughgarden (1979: 5).

The starting point for this work is the assumption that the earliest organisms supported a "large" metabolic map with a "small" number of broad specificity catalysts. General arguments were presented in chapter 2 in support of the

belief that if this were the initial state then high activity, high specificity catalysts can be expected to evolve, given that heritable variation in the determinants of catalyst structure existed and that growth rate was the major (if not sole) measure of fitness in such early organisms.

A number of the specific assumptions used in the model presented in chapter 2 are discarded in chapter 3; in particular, that catalyst activities do not overlap, and that each mutation increases the rate of one catalysed step at the expense of all others (with equal relative effects). This second assumption required the development of a simple abstract view of catalyst-substrate interaction. The model of early organisms that results is still an extremely simple one, for there is no saturation at any metabolic step, and all reactions are monomolecular with first-order kinetics. The environment is homogeneous and contains no other species. The genetics of the cells is also simple: they are asexual clones.

The present work represents an investigation of the evolutionary behaviour of populations of such kinetically simple organisms. The essential approach of the populational aspects has been a *brute-force* one, the machinery of population genetics has not been used. Instead the individual cells, the processes of mutation, growth and cell division have all been directly represented. A single populational concession to practicality has been made: a truncation selection scheme is introduced to increase the relative success of the fastest cells. As the populations being represented have had a fixed size such a concession has increased the chances of progress (higher mean growth rates) by reducing the (nonselective) deaths due to overpopulation.

One of the greatest merits of the approach taken here is that there is a theory of fitness (what it consists in) explicitly built-in to the model. As Bethel (1976), Brady (1979), Gould & Lewontin (1979) and others have argued, the inability to avoid post-hoc "explanations" of relative success undermines the usefulness of natural selection as an explanatory principle. Indeed, in the quotation above, Brady suggests that perhaps we can do without it. The usefulness of natural selection depends upon our ability to identify adaptive traits in organisms "by an engineers criterion of good design, not by the empirical fact of their survival and spread" (Gould, 1976). The complexity of living systems makes this generally impossible to do. The present systems, however, are much simpler to analyse, and the theory of natural selection can be more confidently applied.

The direct representation of cells means that there is no necessity to talk about the average effect of a gene against all backgrounds, so as to assign it a selection coefficient. The model directs its attention exclusively at the fitnesses of cells. The fitness of each cell can be calculated as a function of its total genotype. That function involves a model of enzyme-substrate interaction, and a model of the joint specification of the flux to growth in the cell by the set of protoenzymes each cell contains. As in nature, the fittest cells do not necessarily leave any progeny because the population is finite, and arbitrary events remove cells from it. These calculated fitnesses are explicitly used by the simulation in applying a truncation selection scheme which, given unlimited computing resources, one might prefer to avoid.

A case was presented in chapter 4 that simulation is going to become increasingly important in our understanding of the universe. The complexity of systems and processes in the biological world make this particularly applicable in the life sciences. However, as Conrad has argued, the efficiency with which evolvable systems can be simulated is not great. The present simulation, which takes a simple first-order kinetic system that is subject to evolution and attempts to explicitly represent all its features, is, indeed, demanding on time and resources in its execution. As the cost of processors and memory decreases, and the processing power increases, these demands will become more acceptable, although truly complex evolvable systems will always be prohibitively expensive to model.

In chapter 5, various matters relating to the implementation of scientific programs on computers were discussed. The argument was made that if the complexity of models requiring machine representation is great, then the methods of software engineering, which evolved as techniques for managing complexity, should be used. The use of programming languages which support such methods is highly desirable. It was recommended that the Ada programming language should be used for scientific programming whenever possible. An outline of the specification of the present system in Ada was given, and a few comments on the implementation were made.

Finally, the results of a number of experiments were presented in chapter 6. Several of these experiments involved identical initial populations, and the same metabolic map. In the straight chain pathway experiments no two experiments resulted in the evolution of the same set of activities. The

differences between these simulations are simply due to chance; that is, to the particular genetic events (mutations, duplications and deletions) which occurred at various points of each simulation (and particularly in the early parts before the historical bonds on the population became such that only "fine-tuning" was possible). These events are, the reader will recall, specified by the used of pseudo-random number generators.

In the branched chain experiments briefly discussed at the end of the last chapter, one particular scenario (prohibiting complete blocks in any part of the system) resulted in five out of seven experiments evolving identical sets of protoenzymes in only a few hundred thousand cell divisions. This convergence reminds one of Stebbins (1968: pp 28-30; 1973, chapter 2) discussions on adaptation along "lines of least resistance". Both of the other experiments in the set developed cells with higher growth rates, so the convergence does not represent a global optimum by any means, but a local one that is within easy reach of the initial population In fact, as two variants on the basic solution were found, each more than once, the situation is more complex than this.

What general conclusions might be drawn from this work? Certainly the populations were driven to higher growth rates by selection. In three of the experiments (1, 5 and 6), however, it was necessary to increase the intensity of truncation selection during the experiment because the mean growth rate of the population was not increasing (and in the case of experiment 1, actually decreased) at some point in the experiment. Most of the experiments with a truncation factor of 1.00 suffered population crashes and/or extinction. The

most notable exception was the last experiment, B7, that we looked at. When large-genomed solutions are adopted, the variation in the population has a broader range, and the chances that a few cells can raise the mean sufficiently high that, if they are lost, all cells remaining are below it, is much less.

Higher values for the truncation factor, which reduce truncation selection and hence the artificiality of the simulations, are preferable but have to contend with periods of stasis or retrogression with respect to mean growth rates. The experiments using dispersed storage for the population (experiments 1, A1 and A2) had larger population sizes and did not seem to require high truncation selection, but could not be efficiently sampled. (The non-randomness of the process for selecting cells to be culled from the population largely accounts for this).

As for the initial questions that were asked concerning metabolic evolution, here perhaps the detailed model of enzyme-substrate interaction has interfered with the possibility of direct answers. In these experiments, mono-functionality has not evolved in any experiment. It would have been possible to create scenarios where monofunctionality does inevitably evolve, but not without altering some of the rules of the game.

This was done in a limited way in experiment 6, in which monofunctionality is considerably favoured over multifunctionality, but with the high discrimination Lennard-Jones constants used, the particular route the population took could not have taken it to monofunctionality because of the mutational constraints

imposed. These constraints would not have mattered if the Lennard-Jones values had been those used in the early experiments, because the demand to reach a good fit in one step is not present. A repeat of the experiment with low-discrimination Lennard-Jones constants, therefore, would probably result in the evolution of monofunctional cells. The importance of such an achievement, however, is not great. For the real world, the forces favouring such evolution are, I believe, very great. The qualitative arguments of chapter 2 give my reasons for this belief. Introducing saturation (and hence competitive inhibition) would considerably increase selection for high discrimination enzymes.

Rather amusingly, Koch's (1972) assertions about the structure of protein sequence space turn out to apply to the high discrimination scenarios. It is not always possible to move to a particular functional sequence, in these scenarios, through a series of steps none of which are unfavourable. This was clearly revealed in experiment 6. A refinement on the model, to allow negligible-cost silent sequences which can be reactivated, should be considered for future work. The question of what the structure of protein sequence space is like is, of course, not resolved by this observation. The offending scenario was introduced in the present work not because of problems in the accessibility of sequences in abstract space, but the fitnesses of cells containing the "best" sequences. In the low-discrimination case, the sequences are accessible, but the cells are disadvantaged when considered as systems. In the high-discrimination case with an "overall-fit addition" the cells are at an advantage, but the sequences are not always accessible.

As for the broader issues alluded to in the quotations at the beginning of the chapter, a few points are worth making. The present work has performed evolutionary experiments partly for their own sake. Given a simple organism with a genetic system subject to variation and the simplest of selection schemes, how do populations evolve? In the majority of cases the answer is clear: they converge on some local optimum, to which they become committed (or in which they become trapped if you prefer). Neutral mutations, in the sense of genes and protoenzymes with different sequences but equivalent function, abound but true adaptive polymorphism is rare (not surprising in view of the homogeneity of the environment and the absence of sexual reproduction). To predict the evolution of any given population in any given scenario, it is necessary to know a great deal of background information about it. The constraints on ontogeny and phylogeny that Rosen seeks are indeed not contained within Neo-Darwinism. They depend on particular models of the interactions of significance to the organisms under investigation: models that rest on theories from all areas of biological science.

The core of the problem with Neo-Darwinism is largely that the problem is not with Neo-Darwinism (the fusion of Mendelian and population genetics with Darwinism). The modern synthesis, if it is taken to mean Neo-Darwinism plus the evolutionarily relevant portions of ecology, developmental biology, systematics, and so on (Mayr, 1963), on the other hand, is as unfinished as those bodies of knowledge, and evolves with them. If the synthetic theory is substantially true (and I shall accept, *contra* Laudan, 1977, that truth is the objective of scientific theories) then the relevant material for the questions in which Rosen and Roughgarden are interested must lie in this evolving

material. Of course, the notion that selection is the sole direction giving force in evolution (the synthetic assumption: Bock, 1970) sounds like an empirical statement and so may well be false.

This, I suggest, is one area where the present approach can begin to make an impact in evolutionary theory. For example, critics of the modern synthesis often attack one of the formulations of the synthetic assumption: that sources of variation are random with respect to the direction of evolution (e.g. Webster & Goodwin, 1982; Ho & Saunders, 1984; Reid, 1985; and from the perspective of early biochemical evolution: Fox, 1984). Whatever the randomness of variation at nucleic acid level, the critics say, the consequences of that variation on the phenotype are expressed by physiological and ontogenetic mechanisms that result in decidedly nonrandom effects. Future developments in evolutionary theory, therefore, will pay much closer attention to these *internal factors* (Whyte, 1964; 1965) in determining evolutionary directions. These internal factors, it is essential to note, are not perceived as being the consequence of a past history of selection. They are more to do with the inherent properties of organisms as structurally complex material systems.

(Examples of such effects are not generally provided by the authors, but it is undeniable that complex systems may have many useful features that are not the result of adaptive evolution. For example, there is the demonstration by Kacser & Burns (1981) that the general finding that new mutations are recessive to their wild-type alleles is an inevitable consequence of the nature of metabolic systems, and that no selective explanation is required.

Particular cases of dominance modification may, as Mayo (1983) reminds us, require evolutionary explanation, but the universality of the recessiveness of mutation does not.)

Relatively simple well-founded models of complex systems, forming the basis of evolutionary experiments such as these, can be used to directly address issues of the kind described above. Such models, after all, reflect our current understanding of organisms as complex systems. Outside the synthetic tradition, models of evolution based on the organism are admittedly more anticipated than extant. However, an example is the theory of Brooks & Wiley (1986) which makes assertions about the direction of evolutionary changes in what they argue are entropic properties of populations. Suitable rules for measuring such properties in systems like those described here could thus provide direct examination of that theory. Any view of evolution which makes particular assertions or assumptions about the properties of complex systems can be similarly explored. Within the synthetic tradition, the present approach can be used in the same way, as a test harness for models of particular scenarios in which an explicit description of the organism is required. Only by so doing can the organism be installed in the centre of the evolutionary stage where it surely belongs.

Acknowledgements

No man is an island and this work could never have been completed without the support of others.

I must thank, firstly, Dr Henrik Kacser, for innumerable discussions of great value, for encouragement, and for able supervision. His influence on the shape of the ideas presented here is profound.

The computing required thousands of hours of computer time, and very considerable chunks of my waking hours, neither of which would have been possible without the willingness of my colleague, Mr Iain Richmond, at the Edinburgh Regional Computing Centre, to release me and the machine from other duties.

I thank Dr Jim Burns for discussions early in the work which contributed to the model presented in chapter 2.

I must thank Nahad and Gary Gilbert, who willingly let me use their spare room for a number of weeks in the winter of 1985 when I needed somewhere to work and to house my books, self, etc.

For Yvonne's encouragement and sacrifices, without which this work could never have begun, I am forever grateful.

I must thank Miss Philippa Reed for drawing some pink lines with a highlighting pen on the printouts for experiment 6 (and Miss Fiona Reed for offering to).

Finally, I must thank Meg, whose efforts to provide a pleasant environment for the Amstrad, printouts and me during the writing of this volume made its completion possible.

Appendix

Implementation of Random Number Generators

The Package Random_Integer

```
-- The algorithm used in the following package is based on
-- one given, in a Pascal implementation, by R. Sedgewick (1983)
-- in his book Algorithms

-- The package exports a function which generates
-- random integers in the range 0..Less_Than.
-- Less_Than must lie in the range 0..10000.

package Random_Integer is
    subtype  NATURAL_RANGE is INTEGER range 0 .. 10000;
    function Random (Less_than : NATURAL_RANGE := 100)
       return NATURAL_RANGE;
end Random_Integer;

with Date;
package body Random_Integer is

    T : Date.TIME;
    x : INTEGER;
    Factor : constant := 3141582;
    Max : constant := 100000000;
    SqrtMax : constant := 10000;

    function Mult (p, q : INTEGER) return INTEGER is
        p1, p0, q1, q0 : INTEGER;
    begin
        p1 := p/SqrtMax;
        p0 := p mod SqrtMax;
        q1 := q/SqrtMax;
        q0 := q mod SqrtMax;
        return (((p0*q1+p1*q0) mod SqrtMax)*SqrtMax+p0*q0) mod Max;
    end Mult;

    function Random (Less_Than : NATURAL_RANGE := 100)
       return NATURAL_RANGE is
    begin
        x := (Mult(x,Factor) + 1) mod Max;
        return ((x/SqrtMax) * Less_Than)/SqrtMax;
    end Random;

begin
    T := Date.Get_The_Time;
    x := T.Sec * (1 + T.Min) * (1 + T.Hour);
end Random_Integer;
```

A Published Generic Random Float Package

```
-- The following is a published generic random number generator
-- by B.A.Wichmann and J.G.J.Meijerink
-- "Converting to Ada Packages" in Proceedings of 3rd Joint
-- Ada Europe/AdaTEC Conference

-- The numbers are uniformly distributed

generic
package Gen_Random_Numbers is

    function Random return FLOAT;
    -- Returns a random value in the range 0.0 .. 1.0
    pragma Inline (Random);

    type SEED is
       record
          X, Y, Z : INTEGER;
       end record;

    function  Current_Seed return SEED;

    procedure Restart (Restart_Seed : in SEED);

end Gen_Random_Numbers;
```

```
with Calendar;
package body Gen_Random_Numbers is
   Simple_Case : constant Boolean := INTEGER'Size >= 24;
   -- Simple_Case determines which algorithm will be used,
   -- it should be evaluated by the compiler and in consequence
   -- no additional overhead will be placed on the program

    S : SEED;

    function Random return FLOAT is
       W : FLOAT;
    begin
       if Simple_Case then
          -- Since 30269, 30307 and 30323 are primes, all sequences
          -- can be of maximal length (see Seminumerical Algorithms,
          -- D E Knuth, Addison Wesley 1969, p19).

          S := (X => (171 * S.X) mod 30269,
                Y => (172 * S.Y) mod 30307,
  Z => (170 * S.Z) mod 30323);

       else  -- not Simple_Case
          -- The simple steps above cannot be performed without
          -- overflow on a 16 bit machine.  This is avoided by writing:
          --     Y = k * 176 + r
          -- where   0 <= k <= 172
          -- and     0 <= r <= 175
          -- Then     172 * Y = k * 176 * 172 + r * 172
          --                  = k * 30272 + r * 172
          --                  = - k * 35 + r * 172 mod 30307
          -- Similarly
          -- with           Z = k * 178 + r
          --                170 * Z = - k * 63 + r * 170 mod 30323
          -- and with       X = k * 177 + r
          --                 171 * X = - k * 2 + r * 171 mod 30269
          -- The values are now bounded for a 16 bit machine

          S := (X => 171 * (S.X mod 177) -  2 * (S.X / 177),
                Y => 172 * (S.Y mod 176) - 35 * (S.Y / 176),
                Z => 170 * (S.Z mod 178) - 63 * (S.Z / 178));

          if S.X < 0 then
             S.X := S.X + 30269;
          end if;

          if S.Y < 0 then
             S.Y := S.Y + 30307;
          end if;

          if S.Z < 0 then
             S.Z := S.Z + 30323;
          end if;
       end if;

       W := FLOAT (S.X) / 30269.0 + FLOAT (S.Y) / 30307.0 +
            FLOAT (S.Z) / 30323.0;

       return W - FLOAT (Integer (W - 0.5));
    end Random;
```

```
      ----

      function Current_Seed return SEED is
      begin
         return S;
      end Current_Seed;

      ----

      procedure Restart (Restart_Seed : in SEED) is
      begin
        S    := ( ( Restart_Seed.X - 1 ) mod 30269 + 1,
                  ( Restart_Seed.Y - 1 ) mod 30307 + 1,
                  ( Restart_Seed.Z - 1 ) mod 30323 + 1 );
      end Restart;

      ----

      procedure Initialize_Gen_Random_Numbers is
         use Calendar;
         T     : TIME := Clock;
         Count : INTEGER := INTEGER (Seconds (T) / 3.0);
      begin
         S := (X => Month (T),
               Y => Day (T) + Count,
               Z =>INTEGER (1000.0 *
                   (FLOAT (Seconds (T)) - 3.0 * FLOAT (Count))));
      end Initialize_Gen_Random_Numbers;

   begin
      -- Check that INTEGER has sufficient range
      -- for the generator to work at all.
      if INTEGER'Last < 30323 then
         raise Constraint_Error;
      end if;

      -- Check that Simple_Case gives correct selection.  Raise
      -- Constraint_Error in unlikely case that Simple_Case does
      -- not give correct distinction.
      if Simple_Case and then (INTEGER'Last < 5212632) then
          raise Constraint_Error;
      end if;

      Initialize_Gen_Random_Numbers;

   end Gen_Random_Numbers;
```

References

Abbott, R. J. 1983. Program design by informal English descriptions. *Communications of the ACM* **26**, 882-894

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K. & Watson, J. D. 1983. *Molecular Biology of the Cell.* New York, Garland Publishing.

Andrews, P. R. & Heyde, E. 1979. A common active site model for catalysis by chorismate mutase-phrenate dehydrogenase. *Journal of Theoretical Biology* **78**, 393-403.

Anfinsen, C. B. & Scheraga, H. A. 1975. Experimental and theoretical aspects of protein folding. *Advances in Protein Chemisty* **29**, 205-299.

Anufrieva, E. V., Bychkova, V. E., Krakovyak, M. G., Pauter, V. D. & Ptitsyn, O. B. 1975. A synthetic polypeptide with a compact structure and its self organisation. *FEBS Letters* **55**, 46-49.

ARM. 1983. *The Programming Language Ada Reference Manual.* American National Standards Institute, Inc. ANSI/MIL-STD-1815A-1983. Berlin, Springer-Verlag (one of many publishers).

Barbieri, M. 1981. The ribotype theory on the origin of life. *Journal of Theoretical Biology* **91**, 545-601.

Barbieri, M. 1985. *The Semantic Theory of Evolution.* Harwood, Chur.

Bethell, T. 1976. Darwin's mistake. *Harpers Magazine* **252**, 70-75.

Blake, C. C. F., Grace, D. E. P, Johnson, L. N., Perkins, S. J., Cassels, R., Dobson, C. M., Poulson, F. M. & Williams, R. J. P. 1978. Physical and chemical properties of lysozyme. *Ciba Federation Symposium (New Series)* **60**, 137-172.

Blomberg, C., Ehrenberg, M. & Kurland, C. G. 1980. Free-energy dissipation constraints on the accuracy of enzymic selections. *Quarterly Review of Biophyics* **13**, 231-254.

Blow, D. M., 1978. Flexibility and rigidity in protein crystals. *Ciba Foundation Symposium (NS)* **60**, 55-59.

Bock, W. J. 1970. The synthetic explanation of macroevolutionary change - A reductionist approach. *Bulletin of the Carnegie Museum of Natural History.* **13**, 20-69.

Booch, G. 1982. Object oriented design. *Ada Letters* **1**, 64-76.

Booch, G. 1983. *Software Engineering with Ada.* Benjamin Cummings, Menlo Park.

Brady, R. H. 1979. Natural selection and the criteria by which a theory is judged. *Systematic Zoology* **28**, 600-621.

Brandon, R. N. 1980. A structural description of evolutionary theory. *PSA 1980* **2**, 427-439.

Bremmermann, H. J. 1974. Complexity of automata, brains and behaviour. **In** *Physics and Mathematics of the Nervous System.* M. Conrad, W. Guttinger, & M. Dal Cin (editors). Heidelberg, Springer-Verlag. pp 304-331.

Brocklehurst, K. 1977. Evolution of enzyme catalytic power. Characteristics of optimal catalysis evaluated for the simplest plausible kinetic model. *Biochemical Journal* **163**, 111-116.

Brooks, D. O. & Wiley, E. O. 1986. *Evolution as Entropy.* Chicago, University of Chicago Press.

Bruno, G. 1982. An Ada package for discrete event simulation. *Proceedings ACM AdaTec '82 Conference on Ada.* pp 172-180.

Bruno, G. 1984. Rationale for the introduction of discrete event primitives in Ada. **In** *Simulation in Strongly Typed Languages: Ada, Pascal, Simula.* **13**, no. 2, 10-15.

Bryant, R. M. 1982. Discrete system simulation in Ada. *Simulation* **39**, 111-121.

Buhr, R. J. A. 1984. *System Design with Ada.* Englewood Cliffs, Prentice-Hall.

Burns, J. A. 1971. *Studies on Complex Systems.* PhD Thesis, University of Edinburgh.

Burns, A. 1985. *Concurrent Programming in Ada.* Cambridge, Cambridge University Press.

Burns, A., Lister, A. M. & Wellings, A. J. 1985. *A Review of Ada Tasking.* University of Bradford, Computer Science Report, PR12.

Bychkova, V. E., Gudkov, A, T., Miller, W. G., Mitin, Yu. V., Ptitsyn, O. B. & Spungin, I. L. 1975. Thermodynamic parameters of helix-coil transitions in polypeptide chains. III. Random copolymers of L-leucine with L-glutamic acid. *Biopolymers* **14**, 1739-1753.

Bychkova, V. E., Semisotnov, G. V., Ptitsyn, O. B., Gudkova, O. V., Mitkin, Yu. V. & Anufrieva, E. V. 1980. The compact structure of statistical co-polymers composed of hydrophobic and hydrophilic amino acid residues. *Molecular Biology (USSR)* **14**, 278-286.

Cairns-Smith, A. G. 1971. *The Life Puzzle.* Edinburgh, Oliver & Boyd.

Campbell, M. 1983. Adaptation and fitness. *Studies in History and Philosophy of Science* **14**, 59-65.

Campbell, D., Crutchfield, J., Farmer, D. & Jen, E. 1985. Experimental mathematics: The role of computation in nonlinear science. *Communications of the ACM* **28**, 374-384.

Canovas, J. L., Ornston, L. N. & Stanier, R. Y. 1967. Evolutionary significance of metabolic control systems. *Science* **156**, 1695-1699.

Cantor, C. R. & Jukes, T. H. 1966. The repetition of homologous sequences in the polypeptide chains of certain cytochromes and globins. *Proceedings of the National Academy of Sciences* **56**, 177-184.

Cardelli, L. & Wegner, P. 1985. On understanding types, data abstraction and polymorphism. *Computing Surveys* **17**, 471-522.

Chapman, D. J. & Ragan, M. A. 1980. Evolution of biochemical pathways: Evidence from comparative biochemistry. *Annual Review of Plant Physiology* **31**, 639-678.

Church, A. 1936. An unsolvable problem of elementary number theory. *American Journal of Mathematics* **58**, 345-363.

Clark, B. F. C. 1981. Towards a total human protein map. *Nature* **292**, 491-492.

Cleland, W. W. 1963. The kinetics of enzyme-catalyzed reactions with two or more substrates or products. I. Nomenclature and rate equations. *Biochimica et Biophysica Acta* **67**, 104-137.

Conrad, M. 1974. The limits of biological simulation. *Journal of Theoretical Biology* **45**, 585-590.

Conrad, M. 1983. *Adaptability. The Significance of Variability from Molecule to Ecosystem*. New York, Plenum.

Conrad, M. 1985. Design principles for a molecular computer. *Communications of the ACM* **28**, 464-480.

Cox, B. J. 1986. *Object Oriented Programming: An Evolutionary Approach*. Reading, Addison Wesley.

Creighton, T. E. 1977a. Energetics of folding and unfolding of pancreatic trypsin inhibitor. *Journal of Molecular Biology* **113**, 295-321.

Creighton, T. E. 1977b. Kinetics of refolding of reduced ribonuclease. *Journal of Molecular Biology* **113**, 329-341.

Creighton, T. E. 1978. Experimental studies of protein folding and unfolding. *Progress in Biophysics and Molecular Biology* **33**, 231-297.

Creighton, T. E. 1980. Experimental elucidation of pathways of protein folding and unfolding. **In** *Protein Folding*. R. Jaenick (editor). Amsterdam, Elsevier/North Holland.

Dagley, S. 1975. A biochemical approach to some problems of environmental pollution. *Essays on Biochemistry* **11**, 81-138.

Dayhoff, M. O. 1978. Survey of new data and computer methods of analysis. *Atlas of Protein Sequence and Structure 1978.* M. O. Dayhoff (editor). Volume 5, Supplement 3. New Haven, National Biomedical Press. pp 1-8.

Dean, A. M., Dykhuizen, D. E., & Hartl, D. L. 1986. Fitness as a function of ß-galactosidase activity in *Escherichia coli*. *Genetical Research*. **In press**.

Deisenhofer, J. & Huber, R. 1980. Conformational flexibility and its functional significance. **In** *Protein Folding*. R. Jaenicke (editor). Amsterdam, Elsevier/North Holland Biomedical Press. pp 565-581.

Deutsch, D. 1985. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London, Series A* **400**, 97-117.

Downes, V. A. & Taelleche Bosch, R. 1983. Discrete event modelling in Ada: Implementation and application. **In** *Proceedings of the Third Joint Ada Europe/AdaTEC Conference*. J. Teller (editor). Cambridge, Cambridge University Press. pp 53-63.

Duffy, P. 1971. Studies on control of lactate dehydrogenase activity in mammalian cells I. Demonstration of rapid continuous variations in lactate dehydrogenase activity in cultured cells and their relation to a biological activator and an inhibitor of lactate dehydrogenase. *Biochimie et Biophysica Acta* **244**, 606-612.

Dunbar, R. I. M. 1982. Adaptation, fitness and the evolutionary tautology. **In** *Current Problems in Sociobiology*. King's College Sociobiology Group (editors). Cambridge, Cambridge University Press. pp 9-28.

Edsall, J. T. 1968. Thoughts on the conformations of proteins in solution. **In** *Structural Chemistry and Molecular Biology*. A. Rich & N. Davidson (editors). San Francisco, Freeman. pp 88-97.

Eigen, M. & Winkler, R. 1975. *Das Spiel: Naturgesetze steuern den Zufall*. Munich, R. Piper & Co Verlag.
Translation: *The Laws of the Game: How the Principles of Nature Govern Chance*. Harmondsworth, Penguin Books. 1983.

Eldredge, N. & Gould, S. J. 1972. Punctuated equilibria: An alternative to phyletic gradualism. **In** *Models in Paleobiology*. T. J. M. Schopf (editor). San Francisco; Freeman, Cooper & Co. pp 82-115.

Felhammer, H., Bode, W. & Huber, R. 1977. Crystal structure of bovine trypsinogen at 1.8 Å resolution II. Crystallographic refinement, refined crystal structure and comparison with bovine trypsin. *Journal of Molecular Biology* **111**, 415-438.

Fell, D. A. & Sauro, M. 1985. Metabolic control and its analysis: Additional relationships between elasticities and control coefficients. *European Journal of Biochemistry* **148**, 555-561.

Fersht, A. R. 1974. Catalysis, binding and enzyme-substrate and complementarity. *Proceedings of the Royal Society of London Series B* **187**, 397-407.

Fersht, A. 1977. *Enzyme Structure and Mechanism*. San Francisco, Freeman.

Fersht, A. 1981. Enzymic editing mechanisms and the genetic code. *Proceedings of the Royal Society of London Series B* **212**, 351-379.

Fersht, A. 1985. *Enzyme Structure and Mechanism*. 2nd edition. New York, Freeman.

Fersht, A. R. & Requena, Y. 1971. Equilibrium and rate constants for the interconversion of two conformations of α chymotrypsin. The existence of a catalytically inactive conformation at neutral pH. *Journal of Molecular Biology* **60**, 279-290.

Fischer, E. 1894. Einfluss der Configuration auf die Wirkung der Enzyme. *Berichte der Deutschen chemischen Gesellscheft* **27**, 2985-2993, 3479-3483.

Flint, H., Porteous, D. J. & Kacser, H. 1980. Control of flux in the arginine pathway of *Neurospora crassa*: the flux from citrulline to arginine. *Biochemical Journal* **190**, 1-15.

Flint, H., Tateson, R. W., Barthelmess, I. B., Porteous, D. J., Donachie, W. D. & Kacser, H. 1981. Control of the flux in the arginine pathway of *Neurospora crassa*. Modulations of enzyme activity and concentration. *Biochemical Journal* **200**, 231-246.

Flory, P. J. 1967. Configurational statistics of polypeptide chains. In *Conformation of Biopolymers. Volume I.* G. N. Ramachandran (editor). New York, Academic Press. pp 339-363.

Flory, P. J. 1969. *Statistical Mechanics of Chain Molecules*. New York, Wiley.

Franta, W. R. 1977. *The Process View of Simulation*. New York, North-Holland.

Friel, P. & Sheppard, S. 1985. Implications of the Ada environment for simulation studies. *Simuletter* **16**, 14-26.

Gall, J. 1977. *Systemantics: How Systems Work and Especially Why They Fail*. New York, Times Books.

Garey, M. R. & Johnson, D. S. 1979. *Computers and Intractability*. San Francisco, Freeman.

Ghezzi, C. & Jazayeri, M. 1982. *Programming Language Concepts*. New York, Wiley.

Ghiselin, M. T. 1969. *The Triumph of the Darwinian Method*. Berkeley, University of California Press.

Ghiselin, M. 1974. A radical solution to the species problem. *Systematic Zoology* **23**, 536-544.

Ghiselin, M. 1981. Categories, life, and thinking. *Behavioral and Brain Sciences* **4**, 269-313.

Ghiselin, M. 1985. Can Aristotle be reconciled with Darwin? *Systematic Zoology* **34**, 457-460.

Glasstone, S., Laidler, K. J., & Eyring, H. 1941. *The Theory of Rate Processes.* New York, McGraw-Hill.

Goldschmidt, R. 1940. *The Material Basis of Evolution.* New Haven, Yale University Press.

Goodman, M. M., Newton, K. J. & Stuber, C. W. 1981. Malate dehydrogenase: Viability of cytosolic nulls and lethality of mitochondrial nulls in maize. *Proceedings of the National Academy of Sciences USA* **78**, 1783-1785.

Gould, S. J. 1976. Darwin's untimely burial. *Natural History* **85**, 24-30.

Gould, S. J. 1980. Is a new and general theory of evolution emerging? *Paleobiology* **6**, 119-130.

Gould, S. J. 1982. The meaning of punctuated equilibrium and its role in validating a hierarchical approach to macroevolution. **In** *Perspectives on Evolution.* R. Milkman (editor). Sunderland, Sinauer Associates. pp 83-104.

Gould, S. J. & Eldredge, N. 1977. Punctuated equilibria: the tempo and mode of evolution reconsidered. *Paleobiology* **3**, 115-151.

Gould, S. J. & Eldredge, N. 1986. Punctuated equilibrium at the third stage. *Systematic Zoology* **35**, 143-148.

Gould, S. J. & Lewontin, R. C. 1979. The spandrels of San Marco and the Panglossian paradigm: a critique of the adaptationist programme. *Proceedings of the Royal Society of London Series B* **205**, 581-598.

Groen, A. K., van der Meer, R., Westerfhoff, H. V., Wanders, R. J. A., Akerboon, T. P., & Tager, J. M. 1982a. Control of metabolic fluxes. **In** *Metabolic Compartmentalisation.* H. Sies (editor). New York, Academic Press. pp 9-37.

Groen, A. K., Wanders, R. J. A., Westerhoff, H. V., van der Meer, R. & Tager, J. M. Quantification of the contribution of various steps to the control of mitochondrial respiration. *Journal of Biological Chemistry* **257**, 2754-2757.

Gurd, F. R. N. & Rothgeb, T. M. 1979. Motions in proteins. *Advances in Protein Chemistry* **33**, 73-165.

Kacser, H. 1983. The control of enzyme systems *in vivo*: Elasticity analysis of the steady state. *Biochemical Society Transactions* **11**, 35-40.

Kacser, H. & Beeby, R. 1984. Evolution of catalytic proteins *or* On the origin of enzyme species by means of natural selection. *Journal of Molecular Evolution* **20**, 38-51.

Kacser, H. & Burns, J. A. 1968. Causality, complexity and computers. In *Quantitative Biology of Metabolism*. A. Locker (editor). Berlin, Springer-Verlag. pp 11-23.

Kacser, H. & Burns, J. A. 1973. The control of flux. *Symposia of Society for Experimental Biology* **27**, 65-104.

Kacser, H. & Burns, J. A. 1979. Molecular democracy: Who shares the controls? *Biochemical Society Transactions* **7**, 1149-1160.

Kacser, H. & Burns, J. A. 1981. The molecular basis of dominance. *Genetics* **97**, 639-666.

Karplus, M. & McCammon, J. A. 1981. The internal dynamics of globular proteins. *CRC Critical Reviews in Biochemistry* **9**, 293-349.

Karplus, M. & Weaver, D. L. 1976. Protein folding dynamics. *Nature* **260**, 404-406.

Keightley, P. D. 1982. Enzyme variation and metabolic flux. *BSc Honours Dissertation, Department of Genetics, University of Edinburgh.*

King, J. L. & Jukes, T. H. 1969. Nondarwinian evolution. *Science* **164**, 788-798.

Koch, A. L. 1972. Enzyme evolution. I. The importance of untranslatable intermediates. *Genetics* **72**, 297-316.

Kohn, M. C. & Chiang, E. 1982. Metabolic network sensitivity analysis. *Journal of Theoretical Biology* **98**, 109-126.

Koshland, D. E. 1976. The evolution of function in enzymes. *Federation Proceedings* **35**, 2104-2111.

Kraut, J. 1977. Serine proteases: Structure and mechanism of catalysis. *Annual Review of Biochemistry* **46**, 331-358.

Landauer, R. 1976. Wanted: a physically possible theory of physics. *IEEE Spectrum* **4 (9)**, 105-109.

Laidler, K. J. 1969. *Theories of Chemical Reaction Rates.* New York, McGraw-Hill.

Laidler, K. J. & Bunting, P. S. 1973. *The Chemical Kinetics of Enzyme Action.* 2nd edition. Oxford, Clarendon Press.

Laidler, K. J. & Polanyi, J. C. 1965. Theories of the kinetics of bimolecular reactions. *In Progress in Reaction Kinetics*. volume 3. G. Porter (editor). Oxford, Pergammon Press.

Langridge, J. 1968. Genetic and enzymatic experiments relating to the tertiary structure of β-galactosidase. *Journal of Bacteriology* **96**, 1711-1717.

Laudan, L. 1977. *Progress and Its Problems: Towards a Theory of Scientific Growth*. London, Routledge & Kegan Paul.

Leatherbarrow, R. J., Fersht, A. R., & Winter, G. 1985. Transition-state stabilization in the mechanism of tryrosyl-tRNA synthetase revealed by protein engineering. *Proceedings of the National Academy of Sciences* **82**, 7840-7844.

Levins, R. 1970. Complex systems. **In** *Towards a Theoretical Biology 3. Drafts.* C. H. Waddington (editor). Chicago, Aldine. pp 73-88.

Levinthal, C. 1968. Are there pathways of protein folding? *Journal de Chimie Physique et de Physico-Chimie Biologique* **65**, 44-45.

Lewis, E. B. 1951. Pseudoallelism and gene evolution. *Cold Spring Harbor Symposium on Quantitative Biology* **16**, 159-163.

Lewis, H. R., & Papadimitriou, C. H. 1978. The efficiency of algorithms. *Scientific American* **238**, 96-109.

Lifshits, I. M. & Grosberg, A. Y. 1973. Phase diagram of a polymer globule and the problem of self organization of its spatial structure. *Zhurnal Eksperimentalnoi i Teoreticheskoi Fiziki* **65**, 2399-2420.

Lipscomb, W. N. 1978. Intramolecular interactions, enzyme activity and models. *Ciba Foundation Symposium (NS)* **60**, 1-22.

Liskov, B. & Guttag, J. 1986. *Abstraction and Specification in Program Development*. Cambridge, MIT Press.

Loewy, A. & Siekevitz, P. 1969. *Cell Structure and Function*. 2nd edition. London, Holt, Rinehart & Winston.

Lomow, G. A. & Unger, B. W. 1982. The process view of simulation in Ada. *Proceedings of the 1982 Winter Simulation Conference*. pp 77-86.

Lomuto, N. 1983. Self reproducing Ada tasks. *Ada Letters* **2**, 62-66.

Lovtrup, S. 1974. *Epigenetics: A Treatise on Theoretical Biology*. New York, Wiley.

Lovtrup, S. 1984. Ontogeny and phylogeny. **In** *Beyond Neo-Darwinism*. M. W. Ho & P. T. Saunders (editors). London, Academic Press. pp 159-190.

McLachlan, A. D. 1972. Repeating sequences and gene duplication in proteins. *Journal of Molecular Biology* **64**, 417-437.

McLachlan, A. D. 1980. Pseudo-symmetric structural elements and the folding of domains. **In** *Protein Folding*. H. Jaenicke (editor). Amsterdam, Elsevier/North Holland Biomedical Press. pp 79-99.

Mahan, B. H. 1974. Collinear collision chemistry. *Journal of Chemical Education* **51**, 308-311; 377-380.

Matsuno, K. 1984. Open systems and the origin of protoreproductive units. **In** *Beyond Neo-Darwinism*. M. W. Ho & P. T. Saunders (editors). London, Academic Press. pp 61-88.

Maynard Smith, J. 1961. The limitations of molecular evolution. **In** *The Scientist Speculates*. I. J. Good (editor). London, Heinemann. pp 252-256.

Maynard Smith, J. 1970. Time in the evolutionary process. *Studium Generale* **23**, 266-272.

Mayr, E. 1963. *Animal Species and Evolution*. Cambridge, Belknap Press of Harvard University Press.

Mayr, E. 1970. *Populations, Species and Evolution*. Cambridge, Harvard University Press.

Mazat, J. P., Jean-Bart, E., Rigoulet, M. & Guerin, B. 1986. Control of oxidative phosphorylation in yeast mitochondria - role of the phosphate carrier. *Biochimie et Biophysica Acta* **849**, 7-15.

Maze, J. & Bradfield, G. E. 1982. Neo-Darwinian evolution - Panacea or popgun? *Systematic Zoology* **31**, 92-95.

Michaelis, L. & Menten, M. L. 1913. Die Kinetik der Invertinwirkung. *Biochemische Zeitschrift* **49**, 333-369.

Middleton, R. J. 1980. *Genetic Variation at the* Adh *Locus in* Drosophila. PhD Thesis. University of Edinburgh.

Middleton, R. J. & Kacser, H. 1983. Enzyme variation, metabolic flux and fitness: Alcohol dehydrogenase in *Drosophila melanogaster*. *Genetics* **105**, 633-650.

Mills, S. & Beatty, J. 1979. The propensity interpretation of fitness. *Philosophy of Science* **46**, 263-286.

Nelson, G. & Platnick, N. 1981. *Systematics and Biogeography: Cladistics and Vicariance*. New York, Columbia University Press.

Nelson, G. & Platnick, N. 1984. Systematics and evolution. **In** *Beyond Neo-Darwinism*. M. H. Ho & P. T. Saunders (editors). London, Academic Press. pp 143-158.

Ohno, S. 1970. *Evolution by Gene Duplication*. Berlin, Springer Verlag.

Ornston, L. N. & Yeh, W. K. 1979. Origins of metabolic diversity: Evolutionary divergence by sequence repetition. *Proceedings of the National Academy of Sciences USA* **76**, 3996-4000.

Pace, C. N. 1975. The stability of globular proteins. *CRC Critical Reviews in Biochemistry* **3**, 1-43.

Patterson, C. 1981. Significance of fossils in determining evolutionary relationships. *Annual Review of Ecology and Systematics* **12**, 195-223.

Patterson, C. 1982. Classes and cladists or individuals and evolution? *Systematic Zoology* **31**, 284-286.

Pauling, L. 1946. Molecular architecture and biological reactions. *Chemical Engineering News* **24**, 1375-1377.

Pelzer, H. & Wigner, E. 1932. Velocity coefficient of interchange reactions. *Zeitschrift fur physikalsche Chemie* **B 15**, 445-471.

Pollak, E. & Pechukas, P. 1978. Transition states, trapped trajectories, and classical bound states embedded in the continuum. *Journal of Chemical Physics* **69**, 1218-1226.

Pooley, R. 1985. Languages for discrete event simulation. Simula, Fortran, Pascal, Ada, Modula .....Simula 85? *Proceedings of the 13th Simula Users Conference*. University of Calgary, Alberta. pp 87-92.

Pour-El, M. B. & Richards, I. 1979. A computable ordinary differential equation which possesses no computable solution. *Annals of Mathematical Logic* **17**, 61-90.

Pour-El, M. B. & Richards, I. 1981. The wave equation with computable initial data such that its unique solution is not computable. *Advances in Mathematics* **39**, 215-239.

Privalov, P. L. 1963. [Study of heat denaturation of egg albumin]. *Biofizika* **8**, 308-316. (In Russian).

Privalov, P. L. 1979. Stability of proteins. Small globular proteins. *Advances in Protein Chemistry* **33**, 167-241.

Privalov, P. L. 1982. Stability of proteins. Proteins which do not present a single cooperative system. *Advances in Protein Chemistry* **35**, 1-104.

Ptitsyn, O. B. & Finkelstein, A. V. 1980. Similarities of protein topologies: evolutionary divergence, functional convergence or principles of folding? *Quarterly Reviews of Biophysics* **13**, 339-386.

Ptitsyn, O. B. & Rashin, A. A. 1975. A model of myoglobin self-organization. *Biophysical Chemistry* **252**, 5142-5149.

Quastler, H. 1964. *The Emergence of Biological Order.* New Haven, Yale University Press.

Rajlich, V. 1985. Paradigms for design and implementation in Ada. *Communications of the ACM* **28**, 718-727.

Rapoport, T. A., Heinrich, R., & Rapoport, S. M. 1976. Regulatory principles of glycolysis in erythrocytes *in vivo* and *in vitro* - a minimal comprehensive model describing steady states, quasi-steady states and time dependent processes. *Biochemical Journal* **154**, 449-469.

Reid, R. G. B. 1985. *Evolutionary Theory: The Unfinished Synthesis.* London, Croom Helm.

Richards, E. G. 1980. *An Introduction to the Physical Properties of Large Molecules in Solution.* Cambridge, Cambridge University Press.

Richards, F. M. 1977. Areas, volumes, packing and protein structure. *Annual Review of Biophysics and Bioengineering* **6**, 151-176.

Richards, F. M. & Richmond, T. 1978. Solvents, interfaces and protein structure. *Ciba Foundation Symposium (NS)* **60**, 23-37.

Riedl, R. 1978. *Order in Living Systems.* New York, Wiley. (Originally published in German in 1975 under the title: *Die Ordnung des Lebendigen.* Hamburg, Verlag Paul Parey).

Rosen, D. 1982. Do current theories of explanation satisfy the basic requirements of explanation? *Systematic Zoology* **31**, 76-85.

Rosenberg, A. 1985. *The Structure of Biological Science.* Cambridge, Cambridge University Press.

Rossmann, M. G. 1981. Evolution of glycolytic enzymes. *Philosophical Transactions of the Royal Society of London B* **293**, 191-203.

Rossmann, M. G. & Liljas, A. 1974. Recognition of structural domains in globular proteins. *Journal of Molecular Biology* **85**, 177-181.

Roughgarden, J. 1979. *Theory of Population Genetics and Evolutionary Ecology: An Introduction.* New York, Macmillan.

Salisbury, F. B. 1969. Natural selection and the complexity of the gene. *Nature* **224**, 342-343.

Salter, M., Knowles, R. G., & Podgson, C. I. 1986. Quantitation of the importance of individual steps in the control of aromatic acid metabolism. *Biochemical Journal* **234**, 635-647.

Sander, C. & Schulz, G. E. 1979. Degeneracy of information contained in amino acid sequences: Evidence from overlaid genes. *Journal of Molecular Evolution* **13**, 245-252.

Saunders, P. T. & Ho, M. W. 1981. On the increase in complexity in evolution II. The relativity of complexity and the principle of minimum increase. *Journal of Theoretical Biology* **90**, 515-530.

Savageau, M. A. 1976. *Biochemical System Analysis: A Study of Function and Design in Molecular Biology.* Reading, Addison Wesley.

Schulz, G. E. 1977. Recognition of phylogenetic relationships from polypeptide chain fold similarities. *Journal of Molecular Evolution* **9**, 339-342.

Schulz, G. E. & Schirmer, R. H. 1979. *Principles of Protein Structure.* Berlin, Springer-Verlag.

Sedgewick, R. 1983. *Algorithms.* Reading, Addison-Wesley.

Sheppard, S., Friel, P. & Reese, D. 1984. Simulation in Ada: implementation of two world views. *Simulation in Strongly Typed Languages: Ada, Pascal, Simula* **13**, no. 2, 3-9.

Shnoll, S. E. & Chetverikova, E. P. 1975. Synchronous reversible alterations in enzymatic activity (conformational fluctuations) in actinomyosin and creatine kinase preparations. *Biochimie et Biophysica Acta* **403**, 89-97.

Sober, E. 1984. *The Nature of Selection: Evolutionary Theory in Philosophical Focus.* Cambridge, MIT Press.

Spilsbury, R. 1974. *Providence Lost: A Critique of Darwinism.* London, Oxford University Press.

Stanley, S. M. 1975. A theory of evolution above the species level. *Proceedings of the National Academy of Sciences USA* **72**, 646-650.

Stanley, S. M. 1979. *Macroevolution.* San Francisco, Freeman.

Stebbins, G. L. 1968. Integration of development and evolutionary progress. **In** *Population Biology and Evolution.* R. C. Lewontin (editor). Syracuse, Syracuse University Press. pp 17-36.

Stebbins, G. L. 1974. *Flowering Plants: Evolution Above the Species Level.* London, Edward Arnold.

Steele, S. A. & Beeby, R. 1986. A process simulation package concealing multi-tasking. **In** *Ada: Managing the Transition.* Cambridge, Cambridge University Press. pp 141-152.

Stenseth, N. C. & Maynard Smith, J. 1984. Coevolution in ecosystems - red queen evolution or stasis? *Evolution* **38**, 870-880.

Stern, C. 1960. *Principles of Human Genetics.* San Francisco, Freeman.

Symm, G. T. & Kok, J. 1984. Guidelines for the design of large modular scientific libraries in Ada. In *Proceedings of the Third Joint Ada Europe/AdaTEC Conference, Brussels, 26-28 June, 1984.* J. Teller (editor). Cambridge, Cambridge University Press. pp 153-164.

Symm, G. T., Wichmann, B. A., Kok, J. & Winter, D. T. 1984. *Guidelines for the Design of Large Modular Scientific Libraries in Ada.* National Physical Laboratory Report DITC 37/84 and CWI Note NM-N8401.

Tager, J. M., Groen, A. K., Wanders, R. J. A., Duszynski, J., Westerhoff, H. V. & Vevoorn, R. C. 1983. Control of mitochondrial respiration. *Biochemical Society Transactions* **11**, 40-43.

Tennent, R. D. 1981. *Principles of Programming Languages.* Englewood Cliffs, Prentice-Hall.

Torres, N. V., Mateo, F., Melendez, E. & Kacser, H. 1986. Kinetics of metabolic pathways - a system *in vitro* to study the control of flux. *Biochemical Journal* **234**, 169-174.

Tuomi, J. 1981. Structure and dynamics of Darwinian evolutionary theory. *Systematic Zoology* **30**, 22-31.

Turing, A. M. 1936. On computable numbers, with an application to the Entscheidungs problem. *Proceedings of the London Mathematical Society* **42**, 230-265.

Unger, B. W., Lomow, G. A. & Birtwistle, G. M. 1984. *Simulation Software and Ada.* University of Calgary: Department of Computer Science.

Van Valen, L. 1973. A new evolutionary law. *Evolutionary Theory* **1**, 1-30.

Von Heijne, G., Blomberg, C. & Baltscheffsky, H. 1978. Early evolution of cellular electron transport: molecular models for the ferredoxin-rubredoxin-flavodoxin region. *Origins of Life* **9**, 27-37.

Von Heijne, G. Leimar, O. & Blomberg, C. 1978. On the emergence of new function in primitive proteins. *Journal of Theoretical Biology* **75**, 167-180.

Waley, S. G. 1969. Some aspects of the evolution of metabolic pathways. *Comparative Biochemistry and Physiology* **30**, 1-11.

Wasserman, G. D. 1981. On the nature of the theory of evolution. *Philosophy of Science* **48**, 416-437.

Waters, C. K. 1986. Natural selection without survival of the fittest. *Biology and Philosophy* **1**, 207-225.

Watson, J. D. 1965. *The Molecular Biology of the Gene.* Menlo Park, Benjamin.

Webster, G. C. & Goodwin, B. C. 1982. The origin of species: a structuralist approach. *Journal of Social and Biological Structures* **5**, 15-47.

Wetlaufer, D. B. 1973. Nucleation, rapid folding and globular intrachain regions in proteins. *Proceedings of the National Academy of Sciences USA* **70**, 697-701.

Wetlaufer, D. B. 1980. Practical consequences of protein folding mechanisms. In *Protein Folding*. R. Jaenicke (editor). Amsterdam, Elsevier/North Holland Biomedical Press. pp 323-327 (and peer discussion pp 328-329).

Wetlaufer, D. B. & Ristow, S. 1973. Acquisition of three-dimensional structure of proteins. *Annual Review of Biochemistry* **42**, 135-158.

Wichmann, B. A. & Meijerink, J. G. J. 1984. Converting to Ada packages. In *Proceedings of the Third Joints Ada Europe/AdaTEC Conference, Brussels, 26-28 June 1984*. J. Teller (editor). Cambridge, Cambridge University Press. pp 131-139.

Wicken, J. S. 1980. A thermodynamic theory of evolution. *Journal of Theoretical Biology* **87**, 9-23.

Wicken, J. S. 1984. On the increase in complexity in evolution. In *Beyond Neo-Darwinism*. M. W. Ho & P. T. Saunders (editors). London, Academic Press. pp 89-112.

Wicken, J. S. 1986. Entropy and evolution: ground rules for discourse. *Systematic Zoology* **35**, 22-36.

Wiener, R. & Sincovec, R. 1984. *Software Engineering with Modula-2 and Ada*. New York, Wiley.

Wiley, E. O. & Brooks, D. R. 1982. Victims of history - A nonequilibrium approach to evolution. *Systematic Zoology* **31**, 1-24.

Wirth, N. 1971. The programming language Pascal. *Acta Informatica* **1**, 35-63.

Wirth, N. 1977. MODULA, a language for modular programming. *Software Practice and Experience* **7**, 3-35.

Wirth, N. 1982, 1983, 1985. *Programming in Modula-2*. 1st, 2nd, 3rd editions. Berlin, Springer-Verlag.

Woese, C. R. 1967. *The Genetic Code*. New York, Harper.

Wolfram, S. 1984a. Computer software in science and mathematics. *Scientific American* **251**, 140-151.

Wolfram, S. 1984b. Cellular automata as models of complexity. *Nature* **311**, 419-424.

Wright, P. A., Thomas, J. M., Cheetham, A. K., & Andreas, A. K. 1985. Localizing active sites in zeolitic catalysts: neutron powder profile analysis and computer simulation of deuteropyridine bound to gallozeolite-L. *Nature* **318**, 611-614.

Wu, T. T., Lin, E. C. C., & Tanaka, S. 1968. Mutants of *Aerobacter aerogenes* capable of utilizing xylitol as a sole carbon source. *Journal of Bacteriology* **96**, 447-456.

Wuketits, F. M. 1986. Evolution as a cognition process: Towards an evolutionary epistemology. *Biology and Philosophy* **1**, 191-204.

Wysong, R. L. 1976. *Creation vs Evolution.* San Diego, Master Books.

Ycas, M. 1974. On earlier states of the biochemical system. *Journal of Theoretical Biology* **44**, 145-160.

Yeh, W. K., Davis, G., Fletcher, P., & Ornston, L. N. 1978. Homologous amino acid sequences in enzymes mediating sequential metabolic reactions. *The Journal of Biological Chemistry* **253**, 4920-4923.

Yeh, W. K. & Ornston, L. N. 1980. Origins of metabolic diversity: Substitution of homologous sequences into genes for enzymes with different catalytic activities. *Proceedings of the National Academy of Sciences USA* **77**, 5365-5369.

Yockey, H. P. 1977. Calculation of probability of spontaneous biogenesis by information theory. *Journal of Theoretical Biology* **67**, 377-398.

Zuckerkandl, E. 1975. The appearance of novel structures and functions in proteins during evolution. *Journal of Molecular Evolution* **7**, 1-53.

 From *Ada in Transition* ed. P J L Wallis,
Cambridge University Press, pp 141-152.

A PROCESS SIMULATION PACKAGE CONCEALING MULTI-TASKING

Sarah A. Steele
Richard Beeby
Ada and Software Engineering Technology
59 George Square, Edinburgh, EH8 9JU

Abstract. The paper describes a general purpose simulation package, written in Ada, which makes use of the language's tasking model. It emphasises the suitability of Ada as a language for simulation and highlights the aspects of Ada which made a general discrete event simulation package feasible. The ability to effectively extend the language using general purpose library packages of this kind is seen as one of the benefits of the transition to Ada. The problems encountered in formulating such a package are described, along with their solutions.

INTRODUCTION

This discrete event simulation Ada package is based on the process interaction method of simulation (Franta, 1977). Under this method, the system being modelled is decomposed into processes, each of which describes the behaviour of a specific activity in the real system that evolves concurrently with other activities (see Bruno, 1984).

The package allows the modeller to create three kinds of entity: processes, queues and resources. The queues are represented by lists; resources are also represented by lists with header records defining the maximum and currently available amounts of the resource; and processes are represented by tasks. None of these implementation details are visible to the package user. In particular, the tasking element is completely hidden behind a procedural interface:

```
package Process_Simulation is
   type PROCESS_IDENTITY  is private;
   function Anonymous return PROCESS_IDENTITY;

   generic
      type SIM_TIME    is delta <>;
      type MODEL_ITEMS is (<>);
      with procedure Model_Actions(Model : in MODEL_ITEMS;
                                   Ref   : in PROCESS_IDENTITY);
   package Model_Simulation is
      function  Current_Time return SIM_TIME;
      procedure Set_Current_Time(To : in SIM_TIME);

      function  New_Process(Of_Kind : MODEL_ITEMS;
              Activate_At : SIM_TIME     := Current_Time;
              Creator     : PROCESS_IDENTITY := Anonymous)
         return PROCESS_IDENTITY;
```

```
      type      QUEUE_IDENTITY is private;
      function  New_Queue return QUEUE_IDENTITY;
      function  Queue_For (Name : PROCESS_IDENTITY)
         return QUEUE_IDENTITY;
            -- plus other subprograms for manipulating queues

      type PRIORITY is range 0 .. INTEGER'Last;
            -- higher values have higher priority
      type RESOURCE_IDENTITY is private;

      function  New_Resource(Max_Quantity : POSITIVE)
         return RESOURCE_IDENTITY;
      procedure Seize  (Name     : in RESOURCE_IDENTITY;
                        Caller   : in PROCESS_IDENTITY;
                        Quantity : in POSITIVE;
                        Urgency  : in PRIORITY := 0);
      procedure Release(Name     : in RESOURCE_IDENTITY;
                        Caller   : in PROCESS_IDENTITY;
                        Quantity : in POSITIVE);

      procedure Wait_For_Service(Caller,
                                 Server : in PROCESS_IDENTITY);
      procedure Hold(Name       : in PROCESS_IDENTITY;
                     Interval   : in SIM_TIME);
      procedure Deactivate(Name : in PROCESS_IDENTITY);
      procedure Activate  (Name : in PROCESS_IDENTITY);
      function  Is_Idle   (Name : PROCESS_IDENTITY)
         return BOOLEAN;
      function  Random_For(Entity : PROCESS_IDENTITY)
         return FLOAT;
      procedure Simulate;

      generic
         with procedure Output_Requirements;
      procedure Reset_Simulation;

   private
      ...  -- QUEUE_IDENTITY and RESOURCE_IDENTITY
   end Model_Simulation;
private
   ...  -- representation of PROCESS_IDENTITY
end Process_Simulation;
```

Resource packages in a concurrent environment should generally present a procedural interface (Burns, 1985) and this has the considerable advantage, in this case, that the modeller is not required to know anything about the Ada tasking model (though see later for a complication). Previous process oriented simulation packages by Bruno (1982, 1984) and by Sheppard *et al* (1984; Friel & Sheppard, 1985; based on Bryant, 1982) export tasks visibly, as does a library of packages supporting discrete event modelling in Ada by Downes & Tellaeche Bosch (1984); we feel that the present approach is likely to look less unfamiliar, and less open to misuse.

Ada is a large language that includes many features, the tasking model included, that will be unfamiliar to most software users

with experience of other languages. Where possible, it is clearly desirable that users wishing to take advantage of previously written modules should not themselves require high levels of Ada expertise. A simple interface to a module makes for a more easily used unit, whatever the level of experience the user has.

It is also likely that the more conventional interface will facilitate the transfer of existing simulations in other languages over to Ada. GPSS, Simscript II.5 and Simula have simulation supporting features which are process oriented, with the ability to define both processes and resources. While acknowledging our superficial knowledge of these languages, some preliminary research would appear to support the view that simulations written in these languages could be translated into Ada using the primitives exported by this simulation package, for many applications, almost statement for statement.

These same advantages arise from simply taking non-parallel discrete-event simulation implementations, such as that of the Demos package in Simula (Birtwistle, 1979), and implementing them directly in Ada, as in Lomow and Unger (1982; Inkster *et al.*, 1984). Such an approach, avoiding the use of the concurrency features of Ada, gives a more efficient implementation on existing monoprocessor systems, but at the cost of losing the ability to express true parallelism with potential future gains on multi-processor/distributed systems.

BENEFITS OF USING ADA

We believe that the benefits of using Ada for simulation purposes are very large. Existing first-choice languages for simulation, whatever their merits, do not posses Ada's real-time features, and simulation packages written in them cannot generally be slotted into real-time production systems for such purposes as performance evaluation. The importance of such a facility has been emphasised by Bruno (1984). The potential for use of simulation in design performance management is well illustrated by Gaither (1985).

The use of Ada means, then, that the same language can be used to both control and simulate a system. That is to say, it is possible to test the logical correctness of the actual code for a system, or part of a system, on a simulation model of the system. The possibility of embedding simulation models of critical activities into the real-time management of a system means, we believe, that facilities of the kind offered in a preliminary way by this package should be available in any APSE or IPSE.

THE USE OF Ada CONSTRUCTS

It is worth emphasising here the degree to which the following Ada constructs facilitated the production of a general purpose package:

Generics

Without generic program units, a general purpose module is only possible by providing low level primitives for the user to incorporate into his program. When a concurrent package is being considered, an understanding of Ada's tasking model is clearly required. While the present package does not eliminate the need for the package user to write what will often be a considerable amount of code, the bulk of this will be in a subprogram which is passed as a parameter to the inner generic package. Concurrency is then handled without involving

the user directly (a potential problem here is that the user is effectively able to introduce code into the package, and so potentially undermine its security: see below).

The main package, Process_Simulation, is not generic, but exports a generic package, Model_Simulation, which the user is required to instantiate. There are three formal parameters to this generic package: the first, SIM_TIME, is a fixed point type, allowing the user to specify the absolute accuracy of time measurements and arithmetic; the second is a discrete type, MODEL_ITEMS, the values of which represent the kinds of processes to be modelled; and a procedure, Model_Actions, which encapsulates actions appropriate to each of the values of MODEL_ITEMS. The idea is that the actual subprogram parameter is essentially a case statement which selects for execution the alternative specified by the value of the subprogram's first parameter.

...with tasks as processes

The instantiation of Model_Simulation makes available a function, New_Process, one of the parameters to which is of the discrete type MODEL_ITEMS. A call to this function will (invisibly to the user) create a task, of type PROCESS_TASK, to represent the process (various housekeeping records are also created). The task type specification, hidden in the body of the package, has the form:

```
task type PROCESS_TASK is
   entry Initialise (Model : in MODEL_ITEMS; ...);
   entry Go;
   entry Hold;
   entry Wake_Up;
   entry Shut_Down;
   entry Start_Again;
   entry Service_Completed;
end PROCESS_TASK;
```

Each task is advised of the type of process it is representing through its Initialise entry, so that the correct section of the procedure Model_Actions will be executed by the task.

```
task body PROCESS_TASK is
   My_Type    : MODEL_ITEMS;
   My_Name    : PROCESS_IDENTITY;
   My_Details : REF_ENTITY_RECORD;  -- pointer to record
   ...  -- other declarations
begin
   ...
   accept Initialise (Model   : in MODEL_ITEMS;
                      Name    : in PROCESS_IDENTITY;
                      Details : in REF_ENTITY_RECORD) do
      My_Type    := Model;
      My_Name    := Name;
      My_Details := Details;
   end Initialise;
   ...
   Model_Actions(My_Type,My_Name);
   ...
end PROCESS_TASK;
```

This circumvents the problem that Friel and Sheppard (1985) had of queues only being able to contain processes of one type (requiring that there be as many queue types as modelled process types), since in the present package all processes are associated with a single task type.

Downes & Tellaeche Bosch (1984) did not have the access to a compiler supporting generics when they developed their simulation library and hence had to adopt the technique of using a special task type, with just two entries SIGNAL and WAIT, to sit in the queue instead of the actual process. The drawback of this method is that there is then no way of ascertaining which particular processes are in a queue and no way, therefore, of accessing information about them held in their process-records; nor of aborting these tasks, if necessary, when the simulation ends. Downes & Tellaeche Bosch (1984) left it to the programmer to make sure that all tasks were terminated or aborted once the simulation had come to an end.

Private types

The ability to specify abstract data types is now widely accepted as an essential feature of any programming language for use in large-scale programming. Ada supports such types by private and limited private types, which restrict the available operations to those specified by the user (and, in the former case, to assignment and testing for (in)equality). In the present package, the entities QUEUE_IDENTITY, RESOURCE_IDENTITY, and PROCESS_IDENTITY are all private types, with their representation hidden from the user program. This provides a clean and secure user interface. Appropriate operations for these types are provided, and the user can additionally specify the behaviour of the processes he/she wishes to create.

PROBLEMS ENCOUNTERED

Creating a simulation package that makes use of Ada's multi-tasking model did introduce several problems. For many of these problems, the solutions found have been very satisfactory while others have been less elegantly dealt with.

Scheduling

In existing discrete event simulations, only a single process is active at a time. Process activation involves releasing that process currently at the head of the events list, and updating the simulation time accordingly. The next process is activated whenever the released process returns to the list or terminates.

As Sheppard *et al* (1984) point out, a multi-tasking implementation implies that all processes with the same activation time are actually released together. This creates the problem that the simulation time cannot be advanced again until all currently executing processes have either been resuspended, or terminate. The solution adopted by Sheppard *et al* (to keep a count of the number of processes released, incrementing that count whenever a new process is created and decrementing it as processes complete, deactivate or reschedule themselves) has also been taken here, although it is not secure in the face of failure of any of the released processes (always a possibility, no matter how defensive our programming or powerful our exception handling facilities).

The solution adopted here, however, does simplify life for

the modeller relative to previous solutions:

(a) the responsibility for keeping count is taken by the module, and is hidden from the user (cf. Sheppard *et al*,1984; Downes & Tellaeche Bosch, 1983)

(b) there is no need to report the initial number of processes (cf. Bruno, 1984; Downes & Tellaeche Bosch, 1983; Sheppard *et al*, 1984)

(c) the module imposes no limit on the maximum number of processes in the simulated system (cf. Bruno, 1984).

The administration of these roles falls to a Scheduler task to which the package user has no access. The Scheduler manages the events list upon which processes are suspended to model the passage of time in the simulation. The body of the Scheduler is, in outline:

```
task body Scheduler is
   Active_Tasks : NATURAL := 0;
    ...   -- other declarations
begin
   ... -- start the simulation going
   select
      accept Start do
         -- set appropriate flag
      end Start;
   or
      terminate;
   end select;

   -- the simulation is finished if the (global)
   -- events list is empty
   while Is_Not_Empty (Event_List) loop

      ... -- update simulation time and release all processes
      ... -- having the the same activation time as that of the
      ... -- process at the head of the events list, keeping a
      ... -- count in Active_Tasks

      -- wait for all active tasks to complete,
      -- or reschedule
      while Active_Tasks > 0 loop
         select
            accept Increment do
               -- a new task is being created and
               -- activated immediately or a deactivated
               -- task is being reactivated
               Active_Tasks := Active_Tasks + 1;
            end Increment;
         or
            accept Decrement do
               -- a process is deactivating or joining a queue
               Active_Tasks := Active_Tasks - 1;
            end Decrement;
         or
            accept Reschedule
```

```
                    (Process  : in out REF_ENTITY_RECORD;
                     Interval : in     SIM_TIME) do
                  ... -- place the Process back in the
                  ... -- events list for activation at
                  ... -- (Current_Time + Interval)
                  Active_Tasks := Active_Tasks - 1;
               end Reschedule;
            or
               accept Stop(Name : in PROCESS_IDENTITY) do
                  -- a process has completed
                  Active_Tasks := Active_Tasks - 1;
               end Stop;
            end select;
         end loop;
      end loop;
   end Scheduler;
```

The combination of a concurrent implementation with a central scheduler does create a potential bottleneck on systems with several processors available. The possibility of using distributed scheduling on the lines of Jefferson's (1983; Jefferson & Sowizral, 1983) *time warp* mechanism for implementing *virtual time* is under consideration.

Task Suspension and Resumption

Ada provides three ways of suspending processes without busy waiting. The first of these is the delay statement, which suspends a process for not less than a specified number of seconds. The other two are associated with interprocess communication. If a task issues an entry call on another task it is suspended until the call is accepted and completed. Similarly, if a task reaches an accept statement, and there are no outstanding calls on the entry concerned, it is suspended until such a call is received.

The use of the inbuilt delay statement in the language could not be used to suspend a process for a specified period of simulation time because of the impossibility of knowing what relationship between "real-time" and simulation time obtains for that particular process. Indefinite delays also require to be modelled.

To suspend a process for a definite or indefinite period of simulation time we used entry calls. Each delay involves (though the user is unaware of it) the "creation" of a task, which the process to be suspended calls. The created task is constructed so that it does not accept this call until it has accepted a call that releases it from a queue or wakes it up, depending on the kind of delay involved. In fact, because of the need to have processes of different types held on one queue, the synchronisation task created here is of the same type as the others - PROCESS_TASK - and is accessed via the process' ENTITY_RECORD via a special task-pointer field. The ENTITY_RECORD for each task records whether the task is acting as a synchronisaton task or simulating a process in a Boolean field called Waiting_Process.

```
procedure Hold (Name           : in PROCESS_IDENTITY;
                Delay_Interval : in SIM_TIME) is
   Actual_Record : REF_ENTITY_RECORD := ...;
   ... -- other declarations
```

```
begin
    -- create a synchronisation task, called Dummy,
    -- set the Waiting_Process field to True and
    -- initialise the task
    Dummy                          := New_Process_Task;
    Actual_Record.Waiting_Process  := True;
    Actual_Record.Task_Pointer     := Dummy;
    Dummy.Initialise(Actual_Record.Process_Kind,
                     Name, Actual_Record);
    ...
      -- reschedule the ENTITY_RECORD that now points to the
      -- synchronisation task
    Scheduler.Reschedule(Actual_Record, Delay_Interval);
      -- hold up the main process by making a call on the
      -- synchronisation task
    Dummy.Hold;
      -- once this call has been accepted,
      -- reset relevant record fields and continue
    Actual_Record.Waiting_Process := False;
    Actual_Record.Task_Pointer    := null;
end Hold;

task body PROCESS_TASK is
   My_Details : REF_ENTITY_RECORD;
   ....
begin
        -- initialisation sequence
    select
       accept Go;
          -- from the scheduler
    or
       accept Service_Completed;
          -- from a server process that has finished servicing
    or
       accept Shut_Down do
             -- from the Deactivation procedure
          accept Wake_Up;
       end Shut_Down;
    or
       terminate;
    end select;

    if My_Details.Waiting_Process then
        -- this is a synchronisation task
       accept Hold do
          -- releases the main process task
          -- making the call
          ...
       end Hold;
       ...
    else
       Model_Actions(...);
    end if;
    ...
end PROCESS_TASK;
```

Process Identification

Since there are potentially several processes active concurrently, some or all of which may require the scheduler to specifically reschedule them on the events list, the scheduler must know which ENTITY_RECORD to reschedule. Due to the asymmetry of entry calls in Ada, the identity of a task calling the scheduler is not known. For this reason, each process task is told its identity on creation, and passes this as a parameter to the Scheduler, whenever they call it.

The parameters to the generic package include a subprogram, Model_Actions, which itself takes as a parameter the identity of a process (type PROCESS_IDENTITY). It is therefore clear that such an identity type cannot be exported by the generic package, for the PROCESS_IDENTITY definition must be in scope at the point the generic package is declared, and where it is instantiated. The generic package is thus declared in an enclosing package which also exports the process identity type.

The identification types for queues, resources and processes are private types, and the fact that integers were chosen to represent the values is of no relevance or use to the package user.

Resource Control

Resources, queues and processes are all internally uniquely identified by a number. There is an internal table relating identities to the ENTITY_RECORD representing them; an ENTITY_RECORD being a variant record with fields depending on whether it is representing a process, a queue or a resource.

A resource is actually implemented as a queue, with the header record containing fields for the maximum and current quantities of the resource. Two procedures are provided, Seize and Release, which are parameterised by the kind of resource requested, the amount requested, and the identity of the requesting process. In addition, the Seize procedure has a parameter for specifying the priority of the request (set by default to the lowest value).

Seize will release the process immediately if enough of the resource is available; if not, the process will be deactivated and queued. The queue is ordered on priority, with the priority of a suspended process increasing with time (unless/until it has the highest possible priority). Note that the problems of resource management in Ada, arising out of the resource manager being unable to deactivate calling processes (Wellings *et al*, 1984), are avoided in this package.

At present, there is no facility for seizing different resources simultaneously. We are currently looking into the feasibility of implementing such a facility.

Shared Variables

The problem of shared variables is an interesting one - the more so as the user, since he is unaware of the underlying tasking model, will also be unaware that any variables he declares as global variables are potential shared variables.

The problem does not arise with variables of type PROCESS_IDENTITY, QUEUE_IDENTITY, or RESOURCE_IDENTITY, created using the provided operations, as the package automatically assigns mutual exclusion tasks to protect them. User declared global variables of other types are a problem, however. The only solution to this that

avoids alerting the user to the underlying tasking model (so forcing him to program with concurrency in mind) is to preprocess the user code, attaching mutual exclusion tasks to all global variables and inserting the code to call these tasks wherever the variables are accessed. As any problematic calls will occur in the procedure which is the actual parameter associated with Model_Actions, the amount of work the preprocessor is required to do will generally not be excessive.

Memory Management

The user of the module may declare a new process at any point in his/her simulation. The package, for this and other reasons, creates task dynamically during execution. As the language definition requires that memory allocated to dynamically created tasks cannot be explicitly returned by the user program to the system (through the use of a suitable instantiation of Unchecked-Deallocation, for instance), the tasks are constructed so that, when a particular activation is complete, they do not terminate but return themselves to a pool for reuse (see also Burns, 1985). Tasks are withdrawn from this pool by the internal subprogram New_Process_Task, and created by the language **new** operator only if the pool is empty at the time of the call (and the user program has not reached its system-dependent maximum memory allocation). Without such a pool, the program would eventually, for a long simulation, run out of memory.

```
task body PROCESS_TASK is
   My_Details : REF_ENTITY_RECORD;
   ...
begin
   loop
      ...   -- initialisation sequence
      ...   -- activation sequence
      if My_Details.Waiting_Process then
         select
            accept Hold do
               Return_A_Process_Task(My_Details.Task_Pointer);
            end Hold;
         or
            terminate;
         end select;
      else
         Model_Actions(...);
         Scheduler.Stop(My_Name);
         Return_A_Process_Task(My_Details.Task_Pointer);
      end if;
      ...
   end loop;
end PROCESS_TASK;
```

Determinancy - Repeating Simulations

The major problem with using a multi-tasking model for a simulation package concerns the allocation of processor time to tasks on successive runs of a given simulation. This cannot be explicitly programmed in Ada, and so such runs may result in a different sequence of task activations for each execution of a simulation. As Friel and Sheppard (1985) discuss, this creates problems when, as in virtually all

simulations, a pseudo-random number generator is used, for the order in which processes call the generator, and hence the number they receive, will differ from one run to another. This precludes repeating a simulation, as a level of indeterminancy is admitted into each run.

Friel and Sheppard resolve this problem by giving each process its own random number generator. As processes do not share the generator in this approach, the problem is avoided. Good pseudo-random number generators have long cycle lengths, and are not auto-correlated. It seems likely, therefore, that if such a generator is available, the solution adopted is a sensible one.

We experimented in this package with an alternative solution, which imposes an additional load on the program, but enables all processes to share the same generator. This involves keeping a record of the numbers allocated to different processes, and retaining this information over successive runs. This ensures that the same sequence of numbers is received by each process, whatever the order of the requests is. It may be more statistically sensible not to retain the numbers themselves, but the order in which processes call the generator. It is for repeating simulations that the subprogram, New_Process, has a parameter, Creator, which records the identity of the parent of a given process. The function Anonymous is provided so that first generation processes can be dealt with, and for use in simulations which will run only once.

Statistics Collection

There are currently two schools of thought about statistics collection. One opinion is that statistics collection and reporting should be automatic, as this is an aid to the programmer and more efficient. The other opinion is that automatic statistics collection and reporting is an unnecessary burden on the run-time of the simulation, and that it should be left to the user to do his/her own.

At present, our module exports certain facilities for the user to gain access to statistical information - such as queue lengths etc. However, we are considering the introduction of some basic automatic statistics collection, for the convenience of the user.

IMPLEMENTATION

The work described was performed on a Data General MV4000, using the validated DG/Rolm Ada Compiler (version 2.20 and latterly, 2.30). The availability of a powerful source code debugger was invaluable throughout the development of the package. All of the previous simulation packages we have referred to were hampered by the lack of complete compiler implementations of Ada.

SUMMARY

A general purpose simulation module, requiring of the modeller no knowledge of the advanced features of the Ada tasking model, has been described. The powerful abstraction facilties in the language have been exploited to provide a clean and relatively simple interface to the package user.

The package uses multi-tasking in its representation of processes, and we recognise the inherent inefficiencies of such a multi-tasking package on existing monoprocessor implementations. The representation of an active entity by a task is otherwise sound abstraction, and will be beneficial if future multi-processor

implementations allow true parallelism and are efficient (see, for example, Schonberg & Schonberg, 1985).

The introduction of parallelism, by its very nature, reduces determinancy (see Kuck, 1978), and this has been most strongly seen with the problem of repeating simulations when pseudo-random numbers are required.

Acknowledgements. We are grateful to Rob Pooley for helpful discussions and comments on an earlier draft. We thank a number of anonymous reviewers for comments on an extended abstract of this paper.

References

Birtwistle, G. (1979). DEMOS - A System for Discrete Event Modelling in Simula. New York : Macmillan.

Bruno, G. (1984). Rationale of the introduction of discrete simulation primitives in Ada. In Simulation in Strongly Typed Languages: Ada, Pascal, Simula. 13, no.2, 10-15.

Bryant, R.M. (1982). Discrete system simulation in Ada. Simulation 39, 111-121.

Burns, A. (1985). Concurrent Programming in Ada. Cambridge: Cambridge University Press.

Downes, V.A. & Tellaeche Bosch, R. (1984). Discrete event modelling in Ada: Implementation and application. In Proceedings of the Third Joint Ada Europe/AdaTEC Conference, ed. J. Teller, pp. 53-63. Cambridge: Cambridge University Press.

Franta, W.R. (1977). The Process View of Simulation. New York: North Holland.

Friel, P. & Sheppard, S. (1985). Implications of the Ada environment for simulation studies. Simuletter 16, 14-26.

Gaither, B. (1985). The role of simulation in management of architecture performance. Proceedings of the 13th Simula Users Conference, University of Calgary, Alberta.

Inkster, J., Lomow, G.A. & Unger, B.W. (1984). Combined discrete and continuous simulation in Ada. In Simulation in Strongly Typed Languages 13, no.2, 16-21.

Jefferson, D. (1983). Virtual time. Computer Science Department, University of Southern California, Technical Report TR-83-213.

Jefferson, D. & Sowizral, H. (1983). Fast concurrent simulation using the time warp mechanism. Part I: Local control. The Rand Corporation, Santa Monica, California, Technical Report, June 1983.

Kuck, D. (1978). The Structure of Computers and Computations. 1. New York: John Wiley & Sons.

Lomow, G.A. & Unger, B.W. (1982). The process view of simulation in Ada. Proceedings of the 1982 Winter Simulation Conference. 77-86.

Schonberg, E. & Schonberg, E. (1985). Highly parallel Ada - Ada on an ultracomputer. In Ada in Use, ed. J.G. Barnes & G.A. Fisher, Jr., pp 58-71. Cambridge: Cambridge University Press.

Sheppard, S., Friel, P. & Reese, D. (1984). Simulation in Ada: implementation of two world views. In Simulation in Strongly Typed Languages: Ada, Pascal, Simula. 13, no.2, 3-9.

Wellings, A.J., Keeffe, D. & Tomlinson, G.M. (1984). A problem with Ada and resource allocation. Ada Letters 3, 112-124.

J Mol Evol (1984) 20:38–51

Journal of
Molecular Evolution

Evolution of Catalytic Proteins

or

On the Origin of Enzyme Species by Means of Natural Selection

Henrik Kacser and Richard Beeby

Department of Genetics, University of Edinburgh, West Mains Road, Edinburgh EH9 3JN, Scotland

Summary. It is believed that all present-day organisms descended from a common cellular ancestor. Such a cell must have evolved from more primitive and simpler precursors, but neither their organization nor the route such evolution took are accessible to the molecular techniques available today. We propose a mechanism, based on functional properties of enzymes and the kinetics of growth, which allows us to reconstruct the general course of early enzyme evolution. A precursor cell containing very few multifunctional enzymes with low catalytic activities is shown to lead inevitably to descendants with a large number of differentiated monofunctional enzymes with high turnover numbers. Mutation and natural selection for faster growth are shown to be the only conditions necessary for such a change to have occurred.

Key words: Enzyme evolution — Natural selection — Multifunctional enzymes — Gene duplication — Enzyme specificity — Metabolic evolution

Introduction

There are two principal approaches to research on the early history of life on Earth. The first is forward looking. It is based on experiments and theoretical considerations of abiotic chemistry pertaining to the early conditions on the Earth. The second approach is comparative and retrospective. It attempts to reconstruct the course of evolution by comparing sequences and structures of macromolecules in contemporary species and by considering their distribution among such species.

Theories of the origin of life are possible scenarios about which there is some consensus but also considerable divergence of views. The time interval and changes in conditions between early terrestrial chemistry and present-day organisms is such that no identifiable trace of the former is likely to have been left. By their nature these theories are untestable in any detailed sense and at best are plausible routes of evolution. Their "forward horizon" must end in the early Archean, perhaps 3500 million years ago, since after this fossil evidence shows the existence of recognisable cellular forms.

In contrast, the study of molecular diversity in extant organisms uses unambiguous experimental evidence but has the problem of looking backwards.

The questions which this paper attempts to answer are concerned with the origin of the very large number of diverse enzymes found in contemporary cells. We therefore must examine what light the various retrospective studies shed on this problem.

Comparative biochemistry suggests that the common ancestor of all extant phyla closely resembled a *Clostridium* (Chapman and Ragan 1980). This conclusion implies that this ancestor possessed substantially the same enzyme diversity as that of the modern cell. Questions of the prior evolution of such diversity, therefore, are not answerable by this approach.

The advent of sequencing technology generated the hope that the evolutionary relationships of all macromolecules could be established. Considerable success has been achieved, as exemplified by the unexpected "discovery" of the molecular clock. It seems, however, that there are severe difficulties in establishing relationships between very anciently related molecules using the existing methods of detecting homologies (as will be discussed later).

Comparisons of tertiary structures have been used in arguing about ancestral relationships. For ex-

Offprint requests to: H. Kacser

ample, the nucleotide binding domains of lactate dehydrogenase and glyceraldehyde-3-phosphate dehydrogenase show no detectable sequence homology but are "topologically equivalent" at 57% of their residues (Rossmann and Argos 1977). It has been suggested that natural selection has conserved the three-dimensional structure while allowing sequence to drift (Rossmann 1981). By this argument sequence homology is a poorer guide to divergence between anciently related molecular species than is tertiary-structure homology. The comparison of tertiary structures of proteins to establish evolutionary divergence, however, suffers from the problem that only a limited number of chain folds appear to be thermodynamically favourable. This makes both chance similarity and convergence very difficult to exclude (Ptitsyn and Finkelstein 1980; Richardson 1981; Schulz 1980). Conversely, the absence of any apparent similarity in the tertiary structure of globular proteins generally cannot exclude ancient relationship, as we do not know how extensively tertiary structures may have changed in early evolution if functions diverged. The earliest protein catalysts were likely to have been less compactly folded and have lower stabilization energies than modern enzymes (Flory 1967; Privalov 1979). Descendants of such a common ancestor may well have found different solutions for their folding pathways, particularly if selection was operating to "perfect" them for different coenzymes, prosthetic groups, or substrates. [See also Von Heijne et al. (1978).] Therefore, neither sequence nor tertiary-structure dissimilarity is sufficient ground for excluding common descent from an ancestral molecule if the time of divergence is very distant.

Dayhoff et al. (1978), using computer programs to establish relationships between all published protein sequences, group them into 181 superfamilies among which no meaningful homologies can be asserted. It is conjectured (Dayhoff et al. 1978; Zuckerkandl 1975) that this figure may reach 500 as more data become available. It seems, however, unreasonable to assert that the earliest organisms started "life" with a complement of 500 unrelated proteins. The probability of such a complex system arising spontaneously is too low for it to be considered seriously. The alternative view, that new protein families have been arising de novo throughout evolution, is equally difficult to sustain (although important exceptions like the structurally unorthodox collagens may be cited). We therefore postulate a very much smaller number of proteins in the earliest organisms and must assume that many of these superfamilies are ancestrally related to each other but that sequence homology has been eroded over the aeons.

Such erosion should not be surprising. The detection of sequence homologies between distantly related proteins poses serious difficulties not only because of the operation of the "clock" but because of both insertions and deletions (Dayhoff 1976), which are also a problem for tertiary-structure comparisons (Remington and Matthews 1980; Richardson 1981; Rossmann and Argos 1977). The high rate of superimposed mutations recently revealed by nucleic acid sequencing data (see, for example, Goodman 1981) is a further factor.

It is a feature of arguments in molecular evolution that appeals to natural selection are either post hoc, serving as general explanations for invariant features, or ad hoc, asserting that observed differences are adaptive without, however, specifying the alleged selective pressures. In contrast, our treatment will argue that natural selection is an agent of change and that it can be used predictively in analysing the evolution of the earliest enzyme systems. Our approach is fundamentally different from those described above in that it is essentially an argument based on function and the kinetics of growth. Our paper will be concerned with that period in evolutionary history during which many of the superfamilies, and particularly the enzymes of the metabolic machinery, originated. We place this period between 4000 and 3500 million years ago. In essence our argument is that enzyme systems in early "cells" consisted of a small number of catalytic proteins with low specificity and low turnover numbers, a view previously expounded by Waley (1969), Ycas (1974), and Jensen (1976). We shall investigate the metabolic behaviour of such systems and apply kinetic formulations for their output and growth. We shall establish how the genetically determined kinetic constants are related to growth rate, which must have been the foremost selective parameter in the early cellular phase of evolution. We shall show that selection for growth rate will favour "duplication and divergence" and will inevitably lead to a proliferation of enzyme species. Thus, the "survival of the fastest" will generate the large array of functionally differentiated catalysts which is a feature of all present-day organisms.

Minimal Properties of Early Cells

The study of enzymology and comparative biochemistry reveals two broad facts. First, a high proportion of enzymes in contemporary organisms—no matter what their taxonomic status—are highly specific and considerably efficient as catalysts. Second, the metabolic transformations carried out in all species are essentially the same, even though different solutions to the same problem have been found in some pathways in a few organisms. In fact, there

is a considerable arbitrary element in the biochemical organisation of cells. A clever student of biochemistry could invent a variety of metabolic maps and associated enzymes which would differ substantially from those with which we are familiar. The thermodynamic constraints within which cells operate do not define the particular kinetic organisation that we find. This is a powerful argument for the monophyletic origin of life, or at least for the descent of all present forms from a single type of ancestor. More importantly, it strongly suggests that the evolution of large parts of the metabolic machinery had substantially been completed long before the development of more elaborate forms of life and the divergence of the major groups. Before this common ancestor arrived, there must have been a metabolically less complex precursor, and our starting point will be this primitive cellular or quasi-cellular organism. We shall not be concerned with the evolution of cellular organisation as such prior to that stage. Clearly we must make a number of assumptions about the nature of this early ancestor, since the experimental methodologies discussed in the Introduction give us very little guidance. The assumptions are based on what is reasonably certain about mutation, the nature of catalysis, the thermodynamic and kinetic constraints on reaction chains, and certain properties of "cells" which are generally accepted.

Although we can be certain that these earliest cells were extremely different from our modern cells, they must have had a minimum organisation to be replicating entities. They must have had a metabolism, that is, they must have taken in molecules (and free energy) and transformed them into specific products, including their biocatalysts, the proto-enzymes. They must also have had a genetic material that was replicated with a reasonable degree of fidelity. This must also have had a heterocatalytic activity in influencing the composition and structure, and hence the function, of the proto-enzymes. We shall assume that these enzymes consisted of some form of polypeptides.

Neither the detailed mechanism of replication nor what we now call gene expression need have been exactly like the modern processes. Such cells would have increased in mass and then divided by a process which may, again, have been different from today's. They might, for example, become unstable and divide when the surface-to-volume ratio reached a critical value. Such systems would nevertheless grow exponentially and form a clone.

The Kinetic Structure

While our primitive cell, in its overall behaviour, would therefore have been like modern cells, it is in its detailed biochemical organisation that significant differences would have existed. It is generally agreed that the size of the genome must have been very much smaller. This means that there must have been many fewer "genes" and hence far fewer enzymes dependent on them. The nature of these early proto-enzymes is of critical importance. Modern enzymes, with their high specificity and high turnover numbers, are a small subset of the general class of polypeptides. Mutation studies—both forward and reverse—have revealed the existence of a very large class of mutant derivatives with reduced activities (including some with zero or near zero values for their catalytic functions) and almost none with higher activities. The number of low-activity forms must greatly exceed that of the very special "active" configurations. This suggests that the probability of such high-activity configurations arising by chance is exceedingly low. This is one of the reasons why we believe that modern enzymes have evolved to their present forms from precursors with very much lower activities (Cairns-Smith 1971; Maynard Smith 1961; Ninio 1982). A second and equally compelling reason is that such high activities are unnecessary for sustaining some form of replicating entity. The earliest cells, in their abiotic environment, would be successful with minimal catalytic efficiency so long as the resulting generation times were compatible with avoiding the "thermodynamic death" incumbent upon them due to the finite stability of their molecules.

In spite of the absence of any direct evidence as to the nature of the proto-enzymes, modern enzymology does give us an insight into what they must have been like. Catalysis depends on the interaction of the substrate with the catalyst. This interaction is specified by the free energy of binding with the protein surface. In general, the weaker these interaction energies, the poorer the catalyst will be. It is the free energy of interaction of the transition state between substrate and product that is the relevant measure of the effectiveness of the catalyst. The problem can be treated by transition state theory (Fersht 1977; Jencks 1975), but for our purposes a verbal treatment will suffice (see Appendix II). The free energy of binding can be utilized in lowering the activation energy for the transition reaction. There is a gain due to the enthalpy term and a loss due to the entropy term of the free energy of activation. Because of the relatively weak forces between the transition state and the enzyme, any increase in enthalpy involves a greater participation of the whole surface of the substrate molecules ("multipoint attachment") with consequent greater loss of entropy. In terms of the familiar "lock and key" analogy, the more complementary the enzyme surface is toward one substrate, the higher its cat-

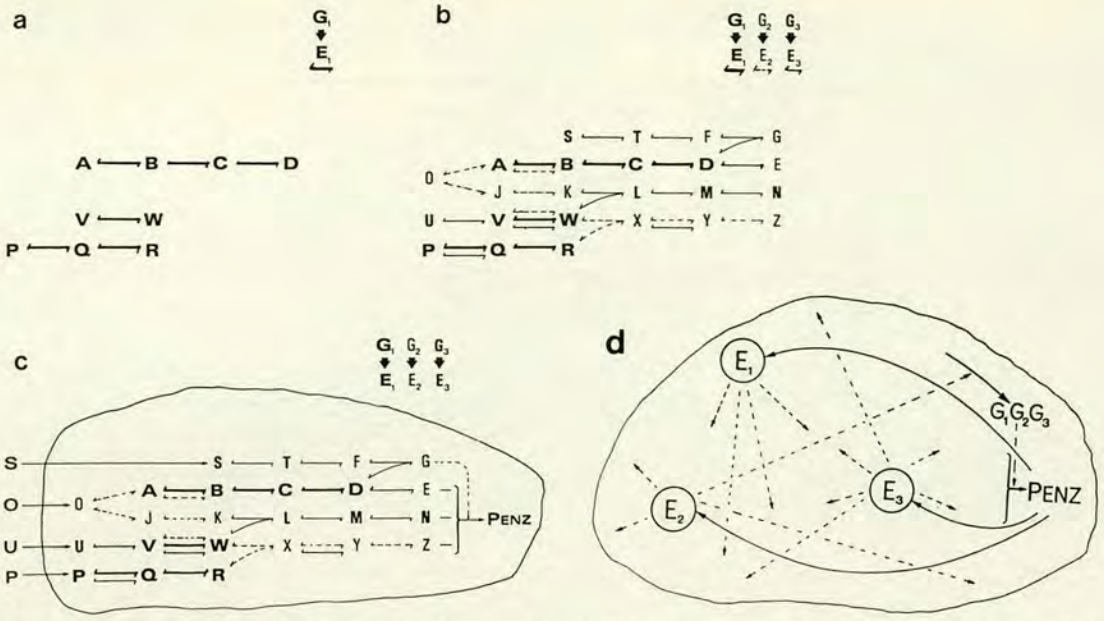


Fig. 1. **(a)** The hypothetical reaction steps shown are catalysed by one enzyme, E_1, which is coded by gene G_1. The reactions are indicated by arrows of equal size, but this does not imply that the catalytic coefficients are equal for all of them. The catalytic effects on all other possible reactions are assumed not to yield reaction rates significantly higher than the spontaneous rate. **(b)** Two more enzymes, E_2 and E_3, with the same type of broad specificity, have been added to the system. The reactions catalysed by the different enzymes are distinguished by the type of arrow. There is some overlap in specificity between the enzymes. **(c)** The complete system is shown, with the supply of external molecular species, S, O, U, and P, and the condensation of products E, N, and Z to form a variety of PENZ, the proto-enzymes, under the influence of G-type molecules indicated. **(d)** A representation of the same system as in (c) showing only the "genes" and enzymes. Not shown, for the sake of clarity, are catalysed steps which do not contribute to PENZ and hence to growth. Such "wasteful" diversion of metabolites would of course occur by the chance properties of proto-enzymes.

alytic rate will be. Conversely, the catalytic activity toward other substrates will be correspondingly lower. It follows that a catalytically poor proto-enzyme will have little discrimination between a range of similar molecules. This broad specificity is not restricted to the substrates but extends to its "function," as the participation of other molecules (co-substrates, cofactors) will be subject to the same imprecision. Water molecules are involved in hydration, hydrolysis, isomerisation, and so forth, and similarly structures containing nucleotides participate in phosphorylations, oxidations, and other reactions (Waley 1969). The proto-enzymes might therefore appropriately be described as "sloppy," as regards both the substrates with which they interact and the reactions that they catalyse. This view of the multifunctional nature of the proto-enzymes generates a very different picture of the kinetic structure of the early cell from that which the "one enzyme–one reaction" view has made familiar. The few enzymes which we had to assume in the beginning now can sustain a relatively large metabolic map—albeit with very poor catalytic effects.

One enzyme might have a range of reactions that it catalyses, as represented in Figure 1a. The individual rates produced by the presence of such an enzyme will, in general, not be the same, but all which are significantly above the spontaneous rate are shown. If two more enzymes are added, each also having a broad range (and some overlaps with E_1), we obtain the possible "map" shown in Figure 1b. The pathways that result, and a large number of others, "existed" prior to the arrival of the enzymes. Their presence allows the kinetic realisation of this particular subset of all thermodynamically possible steps. Another set of enzymes would produce a different map. All these transformations must, of course, originate from the supply of external nutrients and result in the production of proto-enzymes that catalyse the system. We have shown this as a sort of condensation reaction under the influence of other molecules which we call the genetic material (Fig. 1c and d). Once this cycle of events has been achieved, we have a self-sustaining system which will grow exponentially, subject only to the availability of nutrients and the absence of cataclysmic changes in the environment. We shall not address ourselves to the question of whether this was a unique event or many such systems existed, of which our ancestor was the only survivor.

Growth Rate and Fitness

The reasonable degree of fidelity that we assumed in the replication of the genetic material implies a degree of variability. (We need not concern our-

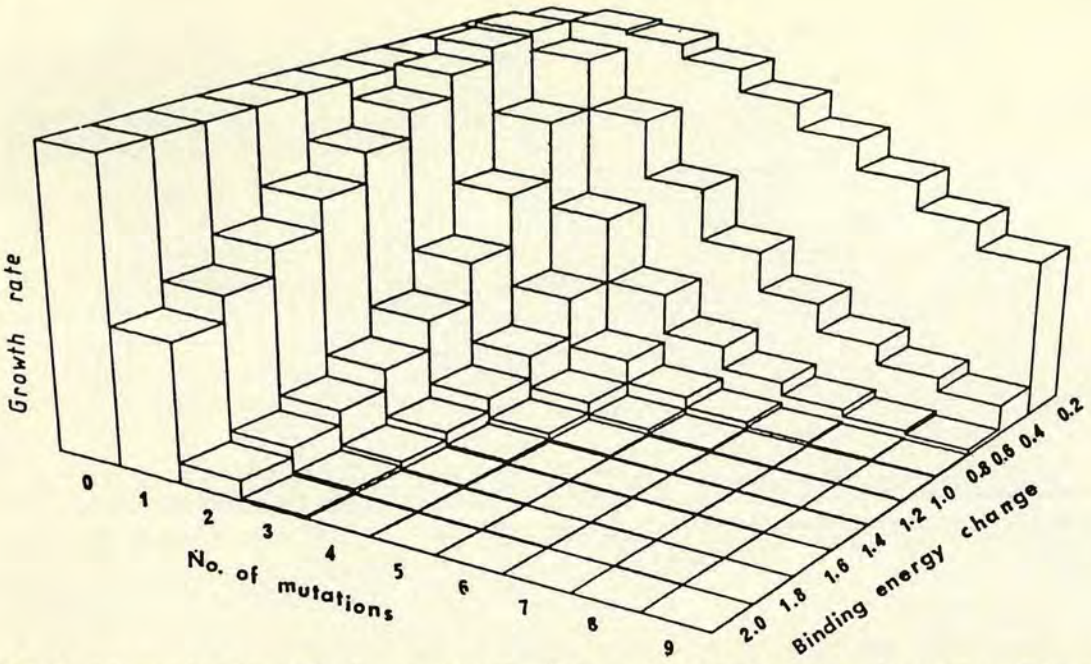


Fig. 2. The mutational trap. The diagram shows the effect on growth of binding-energy changes in a multifunctional enzyme. The subfunction to be improved has a very low catalytic coefficient compared with those of the other subfunctions—the most favourable case for mutational increase in the growth rate. It is seen that only for small binding energy changes is there a marginal increase in growth rate, which, however, declines after one or two such mutations. After enough mutations any mutation "size" will lead to a decline in the growth rate. For details see Appendix II.

selves with the nature of the variation). This must be the source of any evolutionary change. If a cell arose which had an altered growth rate, the resulting clone would either be selected against if it grows slower than or selected for if it grows faster than its neighbors. In time, the one will constitute an ever smaller fraction of the total biomass, the other an ever larger fraction. Since we must assume that in the early phase of evolution there was no sex, no prey, no predators, and no locomotion, almost the only measure of fitness was growth rate. The growth rate of the whole system could increase only by more effective catalysis by its enzymes. Since our proto-enzymes had very low activities, it is an obvious suggestion that mutationlike changes could produce this improvement in catalysis. Consideration of the nature of such changes in the enzymes, however, will show that there are formidable constraints which make increases in the growth rate by mutation alone highly improbable and that this evolutionary route is therefore excluded. The reason is as follows.

The Multifunctional "Trap"

The composition and structure of any proto-enzyme will, of course, be changed by mutations in the genetic material. Such changes will have occurred continuously, probably more frequently than in modern cells with their elaborate error-correcting systems (Haynes and Kunz 1981; Witkin 1976). Like mutational alterations in modern enzymes, many of the changes will tend to disrupt functional integrity, even though there would be quite a number of almost equivalent, poorly effective configurations. Deleterious (and lethal) mutations will generate slower-growing clones, and these would tend to be eliminated, or at best would become a decreasing minority of the population. On the other hand, many configurations with apparently improved catalytic activity will be produced, their original low free energy of interaction having changed to a higher value. This change, however, cannot in general apply simultaneously to all the substrates and functions of one proto-enzyme. We have seen that an increase in rate with respect to one substrate will be generated by an increase in complementarity. In general, the other substrates and functions will have lower interaction energies and suffer a decrease in their rates of transformation (see, for example, Citri and Pollock 1966). Increase in specificity is not an optional "extra" of an improved enzyme but rather a necessary consequence of the mechanism of enzyme catalysis. Since in our multifunctional enzyme mutation will generate more functions with decreased rates than ones with increased rates, the net result will be a decreased contribution to the overall flux of the system. Mutational changes will therefore, on the average, be deleterious. If there are a large number of "wasteful" reaction steps, as indicated in the legend to Figure 1, decrease in these rates will on the average improve the flux to growth. Mutational changes resulting in smaller interaction energies for such steps would be a way to increased growth. The

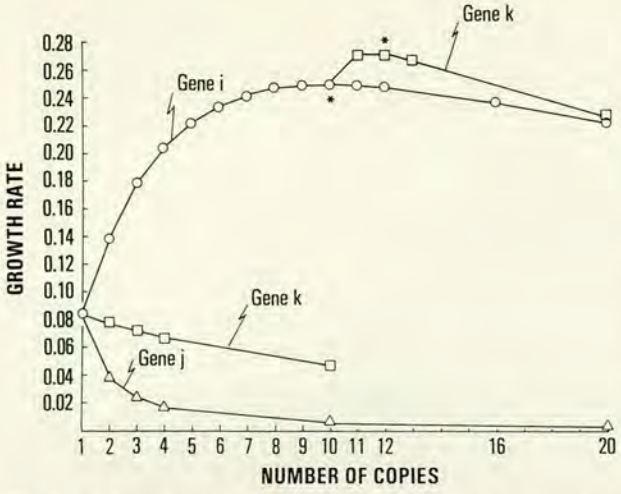


Fig. 3. Effect of gene duplication on growth rate. At the starting position there are 11 genes all occurring as single copies. Growth rates are calculated for three genes with different catalytic coefficients: $c_i = 1$, $c_j = 100$ and $c_k = 20$. Only duplications of *i* result in an increased growth rate; this reaches a maximum at $n_i = 10$. However, if the duplication of *k*, which at the starting position results in a decreased growth rate, takes place after *i* has reached its optimum number, it results in an increase that reaches its optimum at 3 copies. The epistatic interactions inherent in the coupled system have changed the negative coefficient of *k* into a positive one. For details see Appendix I.

strong interdependence of the interaction energies for all the subfunctions, however, puts considerable constraints on the possibility of a net improvement. The cell, having attained the tenuous feat of self-replication by the chance assembly of a few catalytic polypetides, is virtually locked into a growth rate set by the nature of its origin, and can only generate slower clones by mutation (Fig. 2; Appendix II).

There is, however, another mutationlike event which is not subject to the constraints just described. This is an increase in the number of copies of the "gene" in question. Two copies of the gene will have twice the enzyme-specifying capacity. Changes in the amount of enzyme will leave the relative rates of its component functions undisturbed. Whether such increase in the copy number of a gene is brought about by a process analogous to tandem duplication or is due to "sloppy segregation" is irrelevant and we make no assumptions about the mechanism. We shall describe the outcome of either process as "duplication." What is the effect on the metabolism?

The Cost of Allocation

The total amount of polypeptides making up the proto-enzymes is determined by the net output of the metabolic system. How much of this net output is allocated to each individual proto-enzyme will be determined by the relative doses of the respective genes (Flint et al. 1981; Kacser and Burns 1981). (We are assuming that the efficiency of "gene expression" is the same for all genes. Although this is only partly true for contemporary cells, with their complex interactions between genes and gene products, the primitive cell is presumed not to have had such an organisation.) Two doses of one gene will, by increasing the concentration of its dependent proto-enzyme, increase the flux through the steps controlled by it. At the same time, this reallocation of the output will decrease the concentrations of the other proto-enzymes, with consequent decreases in their dependent fluxes. Whether the outcome of these changes in catalysis will be a net increase or decrease in the growth rate is, in general, unspecified. Although each proto-enzyme *concentration* is specified by the allocation, its *catalytic contribution* depends on the thermodynamic and kinetic interactions with its substrates, that is, the catalytic coefficients and the interactions with the rest of the system.

Any such system can be represented by a set of complex nonlinear differential equations for which, however, there is no analytical solution. An algebraic treatment of a simple system can nevertheless lead to conclusions which display the general properties of all catalytic systems (Appendix I). This treatment shows that, on the average, duplication of any one of half the genes will lead to an increase in growth rate, whereas duplication of one of the other half leads to a decrease. For any particular set of quantitative parameters (number of genes, number of dependent functions of each proto-enzyme, differences in catalytic parameters within and between enzymes, structure of the map, etc.), this "50% rule" may not apply. It is, however, certain that for any system there will always be at least one enzyme the duplication of whose gene will increase and at least one for which this duplication will decrease the growth rate. This general answer enables us to predict the evolution of the system in the short term.

Starting with our cell containing one copy of each of the genes, random duplication events will occur in cells of the expanding clone. Those duplications that lead to slower growth are of no interest. Like those cells carrying mutations (discussed above) they will form clones that are on the way to being eliminated. It is the duplications that increase growth which are important. If the duplication events occur at random, such clones will inevitably arise. What fractional improvement is so generated will, again, depend on the particular parameters, but this improvement will in general differ for the different "positive" genes. The one giving the greatest increase is the one most likely to be found with two copies in the increasing population.

If one duplication event can occur, so can a second in that same gene. Will such a third copy give a further increase in growth? Again there is no general answer, but the lower the catalytic contribution of the enzyme was (relative to the others) the more likely it will be that further copies will be selected.

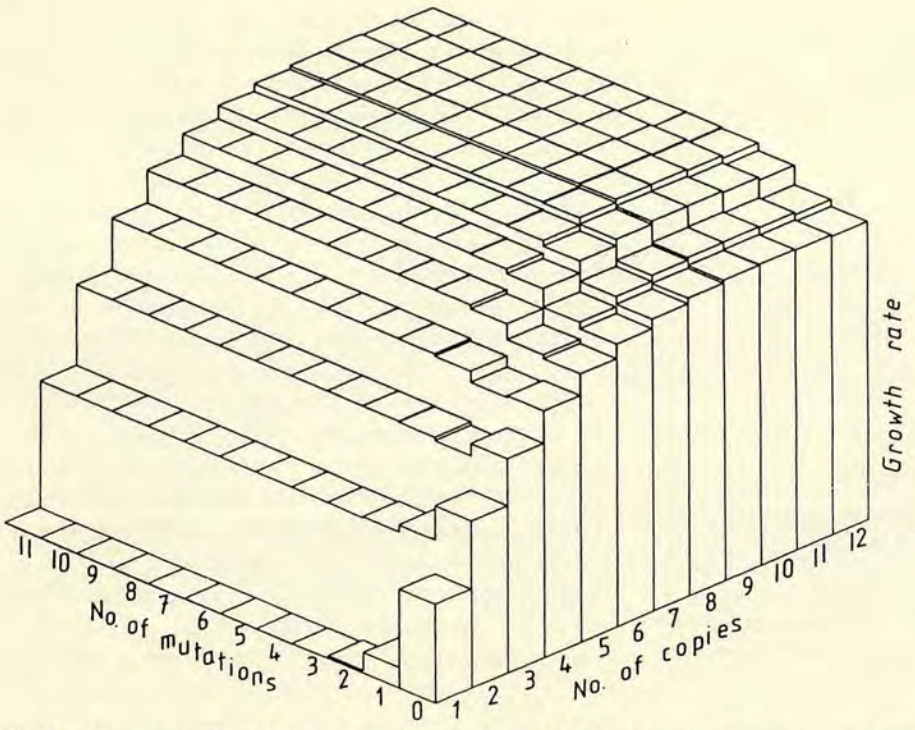


Fig. 4. The escape from the trap. The diagram shows the effect on growth of mutational changes, all involving the same binding-energy change, as a function of copy number in an 11-gene system. In this example, mutation in one copy of a four-copy gene will cause a decline in the growth constant, whereas from five copies onwards the system shows increases in growth rate for the same single mutation. The growth-rate increase due to increase in the copy number at "zero mutation" is the same as given in Figure 3, with a maximum at 10 copies. The binding-energy change was $\delta G/RT = 2.0$ (cf. Fig. 2). For details see Appendix II.

The progress of such a series of duplications, for the case discussed in Appendix I, is shown in Figure 3. It will be noted that the fractional gain for gene *i* declines with the number of copies, the growth rate eventually reaching a maximum (with 10 copies in this example). Each further copy, beyond the maximum-producing number, causes a fractional reduction. Two mechanisms are responsible for this phenomenon. The first is the systemic response of flux to changes in the activity of any one enzyme in a system of coupled reactions (Kacser and Burns 1981). Equal increments in enzyme activity have progressively smaller effects on output. This "law of diminishing return" yields the familiar "dominance curve" in almost every system that has been investigated quantitatively. The second is the progressive effect of allocation of more protein to one enzyme at the expense of the others. Although the increase in growth rate implies an increase in the total available protein, the declining net catalytic effect of each additional copy will progressively increase the "cost" of such allocation. There will come a point—different for different genes—at which the extra copy reduces the rate of some other enzyme more than it increases the rate of its own.

We have now reached a point of balance for duplications of such a gene. At the optimum number of copies, increases or decreases in number will produce slower clones. Natural selection, which previously acted directionally, now will tend to stabilise the system around the optimum number for the gene in question. At this point one of two possibilities for further change can be realised. Since the optimum represents the case in which another enzyme has become catalytically the poorest, the system would gain by increased copy numbers of the locus specifying that enzyme. Random duplication events involving this locus will produce clones which will tend to outgrow all others. This process will continue at almost all loci (with a "reassessment" of all other copy numbers) until a distribution of protein concentrations has been achieved which represents a true maximum growth rate for the system. This optimum allocation depends entirely on the relative catalytic coefficients of the enzymes; a solution for a simple system is given in Appendix I.

The Escape from the Trap

The second possibility depends on the now greatly increased chances that recurring mutations will improve the growth rate. (The algebraic treatment of this situation is given in Appendix II.) The reason for this can be grasped intuitively as follows: duplications in multifunctional enzymes succeed because the relative rates of the subfunctions remain undisturbed, whereas mutations fail because the

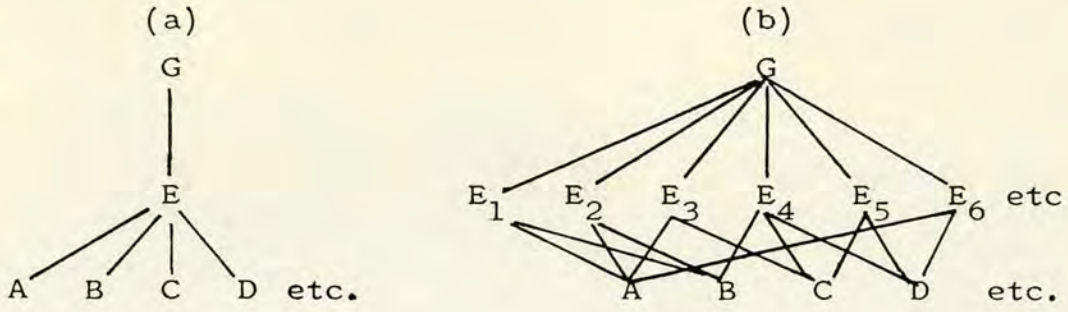


Fig. 5.

changed balance of rates almost invariably reduces the overall flux contribution of the enzyme in question. With multiple copies we have a different situation. Any one of the copies can mutate and leave the remainder with undisturbed specificities toward their product. The presence of such a quantity of product considerably buffers the flux to changes in one copy and allows mutational "adventures" to take place. In fact, some of the same mutations that were previously deleterious will now result in increased growth (Fig. 4; Appendix II). As with duplications, successive mutations in the same copy will increase the growth rate. Unlike the effect of duplications, however, the marginal effect will decline monotonically until the growth rate reaches a plateau value. There will be no optimum since there is no cost of allocation involved. The plateau value is dependent on all other steps, and as changes at these occur, further mutational improvement will be possible. Since the mutational alterations will produce greater complementarity, the resulting enzyme's catalytic coefficient for one subfunction will increase at the expense of those for the others. The way is now opened for the evolution of this enzyme to higher catalytic activity and more restricted specificity—critical features of modern enzymes. In time the catalytic coefficient for the subfunction of the mutation will exceed in contribution that of the remaining copies of the sloppy enzyme. This remainder will therefore be relieved of the necessity to preserve this function, and hence one of these copies will be successful in mutating to improve another of the subfunctions. As before, it will be the function with the lowest catalytic coefficient which is most likely to succeed. As each subfunction becomes autonomous, further mutational improvements become easier. The "shackles are loosened." The outcome of all these changes will be a rapid diversification of enzymes with distinct functions. Starting with a small number of inefficient, multifunctional proto-enzymes the system will end up with a large number of specific enzymes of high catalytic efficiency.

General Considerations

Two mechanisms that may operate in such systems have deliberately been excluded from the simple model for the sake of clarity. They warrant, however, some discussion. They are concerned with the fidelity of gene replication and of "translation."

Mutability is not only a necessary condition for evolution, but also an inherent property of complex molecules. In contemporary cells a whole battery of error-correcting mechanisms has been demonstrated. It must be assumed that these are late refinements which were absent in the early cell. Error-free replication must therefore have been a relatively rare event. Such "sloppy" replication would generate an array of "genotypes." As with modern genes, most such "alleles" would be deleterious or lethal, and the net exponential growth rate of a clone would reflect a balance between "conserved" and "faulty" genomes. Provided the number of conserved progeny cells produced over a period of time is, on the average, greater than one for each parental cell, we would have clonal proliferation of this type no matter what proportion of deleterious genomes is generated.

Similar fidelity considerations apply to the heterocatalytic activity of the genetic material. "Translation" into the proto-enzymes, whatever its mechanism, must be subject to analogous error-prone processes (see Woese 1965, 1972; Ycas 1974). In such a case each gene would produce a family of related peptides with a distribution of catalytic activities, both as regards specificity and efficiency, rather than a single molecular species. While the simple model can be symbolised by (a) of Figure 5, (b) represents, perhaps more realistically, the nature of these early processes. Since peptides of the array would have low affinities, each peptide would have a range of specificities, as in case (a). The different functions, A, B, C, and so forth would therefore be catalysed by a series of related peptides, and the net catalytic coefficient would be represented by a more complex function. Clearly the multifunctional na-

ture of the gene products would be reinforced by such a mechanism. The consequences of this are dealt with in a little more detail in Appendix II. The general conclusions previously derived remain unaffected.

The mechanisms that we have discussed have a bearing on our views of divergence and convergence. Because the structure of each enzyme has been fashioned by requirements for stability and catalytic effectiveness (for which there are probably a limited number of accessible solutions), convergence to similar topologies would almost certainly occur. Therefore, although our model is primarily concerned with divergence from a small number of ancestral molecules, we believe that convergence has also played an important role.

The processes described here as a succession of events would actually, of course, have had considerable overlaps and "false starts." The historical sequence, seen in retrospect, must contain a number of stochastic elements followed by a series of deterministic steps. We can therefore view the "final" modern cell which was the ancestor of all extant forms as having arisen from the chance composition of the self-replicating entity and then evolving by the strict molecular logic of catalysis to the particular and fairly arbitrary kinetic structure of present-day biology. The general direction, however, was inevitable. Thermodynamics and the abiotic environment provided the constraints, the mechanism of catalysis and the kinetics of enzyme systems provided the potential for evolution, but it was natural selection which supplied the driving force.

Acknowledgments. Our thanks are due to Dr. J. A. Burns for helpful discussion.

Appendix I. Allocation Costs of Duplications

Consider a system of enzyme-catalysed transformations whose net output is protein only. The total protein is made up of individual enzyme species whose structural nature is specified by the individual genes. The flux is the rate of output of the sum of all the enzymes, $\sum_{i=1}^{n} E_i$.

Let the flux through this system at steady state be

$$F = f(e_i) \tag{1}$$

where e_i involves enzyme concentrations, E_i (g/ml), as well as their specific catalytic coefficients. The flux, F, is the flux per unit volume (in g/sec/ml). For any given volume, V, the total flux is

$$F_T = V \cdot f(e_i) \tag{2}$$

A change in volume (due to growth), δV, in time δt, gives

$$V \cdot f(e_i)\delta t = \delta V \cdot \Sigma E_i \tag{3}$$

if the output is considered to be enzyme protein only, ignoring any "structural" proteins, which are not included in the flux function. In the limit:

$$1/V \cdot dV/dt = \frac{f(e_i)}{\sum E_i}$$

$$= G, \text{ the exponential growth-rate constant.} \tag{4}$$

We shall examine the behaviour of a straight chain of "unsaturated" enzymes for which algebraic solutions exist (Kacser and Burns 1973, 1981). How far this simplification affects the general conclusions will be discussed later.

The flux for such a linear chain is given by the particular expression for $f(e_i)$:

$$F = \frac{C_x}{\sum_{i=1}^{n} 1/e_i} \tag{5}$$

where C_x is an "environmental" constant, and

$$e_i = \frac{E_i \tau_i}{M_i K_{1i}} \tag{6}$$

where

E_i = enzyme concentration
τ_i = turnover number
M_i = Michaelis constant
K_{1i} = equilibrium constant.

The denominator of expression (5) can be separated into two terms, one concerned with a single enzyme, the other with the rest.

$$F = \frac{C_x}{1/e_i + \sum_{j \neq i}^{n} 1/e_j} \tag{7}$$

since $G = \frac{f(e_i)}{\sum_{i=1}^{n} E_i}$ (or $\frac{F}{\sum E_i}$ for short)

$$G = \frac{C_x}{\sum E_i/e_i + \sum_{j \neq i}^{n} \sum E_i/e_j} \tag{8}$$

Condensing all constants in (6) except enzyme concentration into one, e_i can be expressed as

$$e_i = E_i \cdot c_i \tag{6a}$$

where c_i may be called the catalytic coefficient.

Equation (8) may now be rewritten as

$$G = \frac{C_x}{\frac{\sum E_i}{E_i c_i} + \sum_{j \neq i}^{n} \frac{\sum E_i}{E_j c_j}} \tag{9}$$

The magnitudes of the catalytic coefficient c_i, c_j, etc. are determined by the structures of the enzymes and hence by the natures of the different genes. The magnitudes of the enzyme concentrations, on the other hand, are determined by the relative allocations of the total protein output to each enzyme species. If we assume that the efficiencies of "translation" are the same for all genes, this allocation will be proportional to the fraction of any gene in the genome.

Let n_i = the number of copies (1, 2, 3, . . .) of gene *i*, then

$$\frac{n_i}{\sum_{i=1}^{n} n_i} = \text{fraction of gene } i$$

and $\sum_{j \neq i}^{n} \frac{n_j}{\sum_{i=1}^{n} n_i} = \text{fraction of all other genes}$

Since, in general, the different enzymes will have different molecular weights, m_1, m_2, etc., then

$$\frac{n_i m_i}{\sum_{i=1}^{n} n_i m_i} = \begin{array}{l}\text{fraction of gene } i \text{ product} \\ \text{in terms of grams of protein}\end{array}$$

which is also given by

$$\frac{E_i}{\sum_{i=1}^{n} E_i} \quad \left(\text{or } \frac{E_i}{\sum E_i} \text{ for short}\right)$$

If we designate the molecular weight fraction $\frac{m_i}{\sum m_i} = \alpha_i$, then

$$\frac{E_i}{\sum E_i} = \frac{n_i \alpha_i}{\sum n_i} \tag{10}$$

We therefore have for the terms in equation (9)

$$\frac{\sum E_i}{E_i c_i} = \frac{\sum n_i}{n_i \alpha_i c_i}$$

and

$$\sum_{j \neq i}^{n} \frac{\sum E_i}{E_j c_j} = \sum_{j \neq i}^{n} \frac{\sum n_i}{n_j \alpha_j c_j}$$

We can incorporate the α terms into the values of the catalytic coefficients as a "weighting factor" so c_i from equations (6) and (6a) is now

$$c_i = \frac{\tau_i \alpha_i}{M_i K_{1i}} \tag{6b}$$

and writing $\sum n_i = n_i + \sum n_j$, we have from equation (9)

$$G = \frac{C_x}{n_i + \sum n_j} \times \frac{1}{1/(n_i c_i) + \sum 1/(n_j c_j)} \tag{11}$$

To assess the effect of changing the copy number of one gene on growth while holding the others constant, we rearrange and simplify:

$$G = \frac{L n_i}{n_i^2 + M n_i + N} \tag{12}$$

where $L = \dfrac{C_x}{\sum 1/(n_j c_j)}$

$$M = \sum n_j + \frac{1}{c_i \sum 1/(n_j c_j)}$$

$$N = \frac{\sum n_j}{c_i \sum 1/(n_j c_j)}$$

Similarly rearranging equation (11) gives the dependence of growth on the magnitude of the catalytic coefficient of one enzyme:

$$G = \frac{Q c_i}{P + c_i} \tag{11a}$$

the well-known hyperbolic relationship (Kacser and Burns 1981), where

$$Q = \frac{C_x}{\sum n_i \sum 1/(n_j c_j)}$$

$$P = \frac{1}{n_i \sum 1/(n_j c_j)}$$

How G changes with the number, n_i, depends on the magnitudes of all the parameters. Two useful functions are:

$$\frac{\partial G}{G} \Big/ \frac{\partial n_i}{n_i} = Z_{n_i}^{G} \tag{13}$$

the sensitivity coefficient of growth with respect to the number of copies of one gene, and

$$\frac{dF}{F} \Big/ \frac{de_i}{e_i} = Z_{e_i}^{F} \tag{14}$$

the sensitivity coefficient of the flux with respect to the enzyme activity of the ith enzyme. It can easily be shown that:

$$Z_{n_i}^{G} = Z_{e_i}^{F} - \frac{n_i \alpha_i}{\sum n_i} \tag{15}$$

Clearly, $Z_{n_i}^{G}$ is negative if $\frac{n_i \alpha_i}{\sum n_i} > Z_{e_i}^{F}$, and increases (negatively) as n_i increases. This means that under these conditions G declines, approaching zero as the number of copies, n_i, increases. If $\frac{n_i \alpha_i}{\sum n_i} < Z_{e_i}^{F}$, $Z_{n_i}^{G}$ is positive and G will increase as n_i increases until the two terms in equation (15) are equal, after which G will decline on further increases in n_i. The G versus n_i relationship will therefore show a maximum.

The different parameters in equation (12) will determine which of these solutions obtains in a particular case. We have, however, general expectations. Since it is already known that the sum of all the flux sensitivities, $Z_{e_i}^{F}$, $Z_{e_j}^{F}$, $Z_{e_k}^{F}$, . . . is equal to unity (Summation Property, Kacser and Burns 1979, 1981)

$$\sum_{i=1}^{n} Z_{e_i}^{F} = 1$$

and by definition

$$\sum_{i=1}^{n} \frac{n_i \alpha_i}{\sum n_i} = 1$$

it follows from equation (15) that

$$\sum_{i=1}^{n} Z_{n_i}^{G} = 0 \tag{16}$$

This means that, exploring the behaviour of the growth rate, G, for changes in copy number for all the genes in succession, the sensitivities balance out between positive and negative values. On the average, therefore, about half the genes will increase growth when duplicated, while half will decrease it. The minimum condition, however, is that at least one negative and one positive coefficient be present, the others being constrained by relation (16) (unless they are all individually zero—see later).

We are particularly interested in positive $Z_{n_i}^{G}$ values, since these will form the basis for selecting clones with higher copy numbers. If we start with one copy each of all the genes (as we assume for our early cell)

$$\frac{n_i}{\sum n_i} = \frac{1}{n}$$

where n is the number of different genes in the cell. For positive $Z_{n_i}^{G}$, $Z_{e_i}^{F}$ must be $> \alpha_i/n$ [see equation (15)]. But since the *average* value of $Z_{e_i}^{F}$ is also $1/n$ (Kacser and Burns 1981) it is only $Z_{e_i}^{F}$ values greater than average which will, depending on the enzyme's α_i, give positive growth coefficients. This means that only genes coding for enzymes with catalytic coefficients *lower* than the average are likely to "benefit" from duplication. The worse the enzyme, the greater the gain of multiple copies.

The allocation of the total protein output to the different enzymes via the relative gene fractions therefore carries a "cost" for each duplication. Even when the growth rate increases (positive $Z_{n_i}^{G}$) a relative reduction in the other enzyme concentrations occurs.

The example shown in Figure 3 has been calculated, using equation (12), for the case of 11 different genes starting with one copy each. One gene codes for an enzyme, *i*, having a catalytic coefficient c_i. The average value of the remaining ten c_j's was 100. This gives the parameters L = 10, M = 20, and N = 100 for variation of n_i. The equation for variation in n_j when $c_j = 100$ gives L = 0.92, M = 10, and N = 0.092. The third case uses a value of $c_k = 20$, giving L = 0.95, M = 10.0, and N = 0.48 when at the starting position. A second calculation for *k* starts at the optimal copy number 10 for *i*, giving L = 6.67, M = 19.33, and N = 6.33. All molecular weights are assumed to be equal, and hence all α terms have been disregarded.

We can say immediately that at the maximum ($n_i = 10$), when the total number of genes has increased from 11 to 20, $n_i/\Sigma n_i = 10/20 = 0.5$. From equation (16), $Z_{n_i}{}^G$ must be zero at that point. Therefore the sensitivity coefficient of the *flux* with respect to enzyme activity must also be 0.5. This means that improvement in catalytic activity by mutation is still possible. We shall discuss the conditions for this in Appendix II.

The second result seen by these examples is that genes with enzyme activities much greater than c_i have negative $Z_n{}^G$ values, thus generating slower growing clones. The situation, however, changes when n_i has reached its maximum of 10 copies. At that point the reallocation of protein has reduced the concentrations of each of the other enzymes from 1/11 (0.091) to 1/20 (0.05). A further copy of *i*, increasing the net activity, n_ic_i, by a small amount (because of the decreasing effect of further increase in activity on the flux), reduces the net activity of the other enzymes more than the marginal gain due to *i*. This means that another enzyme has become the "worst," displacing *i* from that position. Increase in copy number of this gene, therefore, will result in further increase in growth. Gene *k*, in our example, is such a gene, as can be seen by the increase if duplication of *k* starts at $n_i = 10$. Its optimum is reached with three copies.

Depending on the distribution of activities, a number of such "second round" duplications can occur. Furthermore, "third round" duplications will be possible for those genes that have already reached their "first optimum." Thus, in the example given, once n_k has reached 3, n_i is no longer at the optimum with 10 copies, but will produce improvement in growth by a further three copies, giving G = 0.279. This pulling itself up by its own bootstraps, however, will come to an end because of the constraints of relation (16). One such situation occurs when all the growth sensitivities individually are zero. Equation (15) shows that at this position *any* change in the number of copies must produce some coefficient which is negative, i.e., will move away from its maximum. We therefore have reached an optimum allocation of protein. (Since copy numbers are not infinitesimally variable but rather change by integers, this will not correspond exactly to a maximum growth rate, but will be the nearest possible allocation of the relative enzyme concentrations. All $Z_{n_i}{}^G$'s will therefore be approximately zero.)

We can determine what this optimum allocation is. At $(Z_{n_i}{}^G)_{max}$ we find from equation (15) that

$$Z_{e_i}{}^F = \frac{n_i\alpha_i}{\sum n_i} = \frac{E_i}{\sum E_i} \quad \text{[from relation (10)]}$$

The formulation for $Z_{e_i}{}^F$ (Kacser and Burns 1973, 1981) is given by

$$Z_{e_i}{}^F = \frac{1/e_i}{\sum 1/e_i} = \frac{1/E_ic_i\alpha_i}{\sum 1/E_ic_i\alpha_i}$$

It follows that

$$E_i{}^2 = \frac{1}{c_i\alpha_i} \cdot \frac{\sum E_i}{\sum 1/E_ic_i\alpha_i}$$

or $$E_i \propto \frac{1}{\sqrt{c_i\alpha_i}}$$

The optimum allocation is therefore given by

$$E_i{:}E_j{:}E_k{:}\ldots = \frac{1}{\sqrt{c_i\alpha_i}}{:}\frac{1}{\sqrt{c_j\alpha_j}}{:}\frac{1}{\sqrt{c_k\alpha_k}}{:}\ldots$$

Further progress by duplication alone will therefore stop. Mutational changes will now be dealt with in Appendix II.

Appendix II. Mutations in Multifunctional Enzymes

The model system of Appendix I consisted of a number of gene-dependent enzymes, each catalysing *one* step in a chain of "linear" enzymes. This is an unnecessary restriction. Since the flux function [Equations (5) and (6a)] contains a sum of terms, they can be grouped in sets such as:

$$\sum 1/E_ic_i = (1/E_1c_1 + 1/E_2c_2 + 1/E_3c_3) + (1/E_4c_4 + 1/E_5c_5) + \ldots$$

If each group is catalysed by the same protein, the denominator of equation (5) can be rewritten as

$$1/E_i(1/c_1 + 1/c_2 + 1/c_3) + 1/E_j(1/c_4 + 1/c_5) + \ldots = 1/(E_ic_i{}^*) + 1/(E_jc_j{}^*) + \ldots$$

where $$1/c_i{}^* = \sum_{i=1}^{3} 1/c_i$$

$$1/c_j{}^* = \sum_{j=4}^{5} 1/c_j$$

etc.

This is the formulation which was used in equations (7) and (9), where all terms were separated into two, one concerned with one enzyme only and one with the sum of the rest. The analysis therefore applies equally to multifunctional enzymes. Furthermore, the sequence of the steps is irrelevant since the sum in equation (5) is independent of order. A particular protein can therefore be considered to catalyse a number of steps straddling the whole sequence.

We shall again divide the system into two groups, one whose steps are catalysed by one enzyme and whose mutational behaviour we wish to investigate, the other the sum of all the other groups, which remains constant. Thus, equations (11) can now be written as

$$G = \frac{C_x}{n_i + \sum n_j} \cdot \frac{1}{1/n_i(1/c_A + 1/c_B + 1/c_C + \ldots) + S} \qquad (17)$$

where the subscripts A, B, C, ... of the catalytic coefficients signify the particular functions catalysed and S is the last Σ term of equation (11). The different functions (steps) will, in general, have different associated magnitudes of catalytic coefficients since they will not only involve the interaction of one protein with different substrates and products, but will also have different equilibrium constants, as seen in relation (6). We now wish to ask what effect a mutation has on the magnitudes of the various catalytic coefficients of the group.

Although a mutation which increases the magnitude of a catalytic coefficient of a monofunctional enzyme will always lead to an increase in growth rate [Eq. (11a)], this is not necessarily the case for a multifunctional enzyme.

The ratio τ_i/M_i from relation (6) (which is usually designated by k_{cat}/K_m in kinetic publications) is the effective rate constant for the formation of the enzyme–substrate complex. It is likely that the energetics of the formation of the substrate–product transition state are rather more important than the undisturbed substrate itself. The role of the free energy of activation for the process is given by the usual formulation

$$\frac{\tau_i}{M_i} = \frac{kT}{h} e^{-\Delta G^*/RT}$$

The free energy, ΔG^*, can be subdivided into two terms $\Delta G_R{}^*$, the activation energy for the chemical transition, and ΔG_B, the binding energy of the enzyme with the transition state.

We can therefore write:

$$\frac{\tau_i}{M_i} = \frac{kT}{h} e^{-\Delta G_R{}^*/RT} e^{-\Delta G_B/RT}$$

$\Delta G_R{}^*$ is positive, ΔG_B negative. We shall assume that changes in enzyme configuration will principally affect the binding energy insofar as the groups on the enzyme form various types of bonds (hydrogen, van der Waals, polar, etc.) with various groups or parts of the substrates, whereas the chemical transition is relatively unaffected by such changes. The binding energy, however, plays a vital role in the net catalysis, since it is "utilized" in lowering the overall activation energy. We can now write

$$\frac{\tau_i}{M_i} \simeq C e^{\Delta G_B/RT}$$

where C represents the (unchanging) "chemical" terms and the exponential term has taken account of the algebraic sign. Any change in enzyme configuration due to mutation will therefore alter the binding energy by δG_B, giving a new value $(\tau_i/M_i)_m$ for the mutant enzyme.

$$\left(\frac{\tau_i}{M_i}\right)_m \simeq C e^{(\Delta G_B + \delta G_B)/RT}$$

$$= \frac{\tau_i}{M_i} e^{\delta G_B/RT} \quad \left(\text{or } \frac{\tau_i}{M_i} e^{\delta} \text{ for short}\right)$$

The change in the rate constant is therefore directly related to the change in binding energy. We are interested in increases in binding energy, i.e., in changes which will increase the rate of transformation. Such changes will inevitably increase the "complementarity" between the two partners in the interaction. A corollary of this is that the complementarity between the enzyme and another substrate, which differs in some way from the first, will be decreased, i.e., the δG_B for this reaction will have the opposite sign. The increase in the rate constant for one substrate will therefore be accompanied by a decrease in the rate for the other substrates. The relative magnitudes of these positive and negative changes in binding energy will depend in a complex fashion on the particular substrates involved and the changes in the groups on the enzyme. As a first approximation, and for algebraic convenience, we shall make the simplifying assumption that they are equal and opposite. Our general conclusions from the subsequent analysis are, however, not dependent on this condition.

The terms for the catalytic coefficients in our growth equation (17) contain not only τ_i/M_i but also the equilibrium constants K_{1i} between substrates and products and α_i, the molecular weight fraction [Eq. (6b)]. These are however, unaffected by any changes in catalysis. We can therefore write

$$(c_i)_m = c_i\, e^{\delta} \tag{18}$$

or

$$(1/c_i)_m = 1/c_i \cdot e^{-\delta} \tag{18a}$$

and for a set of steps catalysed by the same enzyme we will have one "improved" coefficient and the remainder reduced.

$$\left.\begin{aligned} (1/c_A)_m &= (1/c_A)e^{-\delta} \\ (1/c_B)_m &= (1/c_B)e^{\delta} \\ (1/c_C)_m &= (1/c_C)e^{\delta} \end{aligned}\right\} \tag{19}$$

etc.

The changed growth rate equation for the system with the new mutant enzyme will therefore be for the case of a single copy ($n_i = 1$)

$$(G)_m = \frac{C_x}{\sum n_i} \cdot \frac{1}{(1/c_A)e^{-\delta} + \sum_B^n (1/c_B)e^{+\delta} + S} \tag{20}$$

and for multiple copies of gene i

$$(G_{n_i})_m = \frac{C_x}{n_i + \sum n_j} \cdot \frac{1}{1/c_A\left(\frac{1}{n_i - 1 + e^{+\delta}}\right) + \sum_B^n 1/c_B\left(\frac{1}{n_i - 1 + e^{-\delta}}\right) + S} \tag{21}$$

Two different conditions apply for mutational improvement by a change e^{δ} for the two cases (20) and (21):

(a) *Single gene copies*

Increase in growth rate [$(G)_m > G$] will occur if

$$(1/c_A)e^{-\delta} + \sum_B^n 1/c_B e^{+\delta} < 1/c_A + \sum_B^n 1/c_B$$

This simplifies to

$$e^{-\delta} > \frac{\sum 1/c_B}{1/c_A} \tag{22}$$

Since $e^{-\delta}$ is always <1, a necessary condition is that:

$$1/c_A > \Sigma\, 1/c_B \tag{23}$$

and, depending on the value of δ, perhaps

$$1/c_A \gg \Sigma\, 1/c_B \tag{23a}$$

This means that only functions with catalytic coefficients very much smaller than the average of the rest of the coefficients would be subject to mutational improvement. Alternatively, we can say that for a given set of c values only δ changes which are small will be effective. By the same argument such improvements will be very small. All other δ changes will result in a decrease in growth rate.

Assuming that such an increase in growth rate has taken place consequent upon an appropriate δ change caused by a structural change in the enzyme, the new values for the coefficients, given by relation (19), will have produced a decrease for $(1/c_A)_m$ and an increase for all the other $(1/c_B)_m$'s. The ratio $\Sigma\, 1/c_B : 1/c_A$ may have become >1. In such a case there will be *no* value of δ which could produce a further increase in the growth rate. The enzyme is therefore "trapped" in the configuration it has and can only produce variants that will be at a selective disadvantage. This was the condition used in the descriptive text.

(b) *Multiple gene copies*

The condition for improvement is, however, very much less stringent if a gene has undergone duplications. From equation (21) the condition for $(G_{n_i})_m > G_{n_i}$ is

$$1/c_A\left(\frac{1}{n_i - 1 + e^{+\delta}}\right) + \sum 1/c_B\left(\frac{1}{n_i - 1 + e^{-\delta}}\right) < 1/c_A \cdot 1/n_i + \sum 1/c_B \cdot 1/n_i$$

This simplifies to

$$\frac{n_i - 1 + e^{-\delta}}{(n_i - 1)e^{-\delta} + 1} > \frac{\sum 1/c_B}{1/c_A} \qquad (24)$$

While $e^{-\delta}$ is always <1 and therefore severely limits the conditions for improvement in single copies, the function in relation (24) is always >1 for values of $n_i > 2$ and increases with the number of copies. The number of functions which can be mutationally improved is therefore very much greater and an enzyme "trapped" (as a single gene copy) becomes "liberated" for mutational improvement. As a corollary much larger binding-energy changes are now accessible. As before, $(\Sigma\, 1/c_B)/(1/c_A)$ will increase with mutation for increases in c_A, which, in contrast to the single-copy situation, will allow further mutational increases to take place. There will *always* be some changes which result in an increased G value.

The net result is thus a progressive change in the catalytic activity *and* specificity for function A with decrease (and eventual loss) for the other functions. The enzyme, originating from one of the multiple gene copies, is therefore on the way to monofunctionality. The only limits are the possible structural alternatives that mutations can bring about in the complementarity of the enzyme surface.

The examples given in Figures 2 and 4 have been calculated for a succession of mutational steps of equal binding-energy changes. Equation (21) becomes

$$(G_{n_i})_m = \frac{C_x}{n_i + \sum n_j} \cdot \frac{1}{1/c_A\left(\frac{1}{n_i - 1 + e^{+\delta m}}\right) + \sum 1/c_B\left(\frac{1}{n_i - 1 + e^{-\delta m}}\right) + S} \qquad (25)$$

where $m = 0, 1, 2, \ldots$ equals the number of mutations with free binding-energy changes, $\delta G/RT = \delta$.

Figure 2 shows the change for a single copy ($n_i = 1$) for the case where $\Sigma\, 1/c_B : 1/c_A = 0.43$, i.e., a condition which allows some mutational improvement for small values of δ. These were chosen as 0.2, 0.4, 0.6, . . . , 2.0 and nine successive mutational steps are shown for each value of δ. After enough mutations any mutation "size" will lead eventually to a decline in growth rate.

Figure 4 is based on the calculation using a single value of $\delta = 2.0$ with $m = 0, 1, 2, \ldots, 11$ and varying the number of copies, n_i, from 1 to 12. Up to four copies are seen to lead to a decline, but with larger numbers the effect begins to reverse and mutational improvements take place with the growth rates tending towards plateau values. The increase due to copy numbers only ($m = 0$) is the same as that shown in Figure 3.

Error-Prone Translation

The algebraic arguments above were developed for a multifunctional single enzyme specified by one gene. There are, however, good grounds for believing that early "translation" mechanisms were error prone, which would result in the production of a family of related polypeptides by the same gene. Duplications of such genes would have precisely the same effects as were described for the simpler model. Mutations in such a system, however, would have effects additional to those discussed. We are not concerned with those mutations that result in a new array of enzymes which fails to catalyse one or more of the functions. These would be lethal. Some mutations might cause the probability of some peptides to be produced to be shifted at the expense of others. This is equivalent to a change in the concentration terms for the respective functions. For example, for function A in Figure 5b of the text the relevant terms might be

$$[1/E_1c_1 + 1/E_2c_2 + 1/E_3c_3 + 1/E_6c_6]_A + \ldots$$

A change in concentration of the E terms may increase or decrease the net A rate, but there will be corresponding changes in the function B, C, etc. terms. It is unlikely that such random changes will result in an improvement of all the functions. Should it occur, the gain would be marginal, and with a further mutation of this kind, the system would find itself in the trap already discussed. Mutations resulting in one or more peptides of the array gaining a higher affinity for one of the substrates would run into the same difficulties of reduced binding to other substrates already discussed. A third type of mutation could increase the fidelity of translation, by whatever mechanism. Insofar as such a mutation might reduce the production of peptides without any catalytic activity, there would be a net gain and hence positive selection.

It would be unprofitable to attempt an algebraic treatment of this situation since the number of arbitrary assumptions necessary would make it impossible to draw any general conclusions. Intuitively, it seems likely that the existence of multiple peptide species only decreases the probability of a favourable mutation occurring in a single-copy situation. Multiple copies, on the other hand, will, as shown before, open up the system to progress by mutation. We therefore believe that the existence of error-prone processes introduces no new properties to the behaviour of the system except in the magnitude of its parameters.

Other Deviations

Further aspects of the simplifications employed in the model will now be considered. The straight chain of unsaturated enzymes was chosen as a convenient algebraic device having analytical solutions. The restriction to a straight chain is clearly not necessary. Any set of linear equations resulting from more complex reaction schemes would be soluble, but the algebraic expressions for the solution would soon become very dense indeed, and we would loose a great deal of immediate insight into their properties. If saturation and bimolecularity are introduced, the resulting set of nonlinear equations has no analytical solutions. It has been shown, however, that the Summation Properties are quite general and apply to all sets of equations (Kacser and Burns 1979). The principal conclusions concerning duplications and growth therefore hold, distinct from the particular examples calculated for demonstration purposes.

A similar argument applies to the effect of mutations. The inversion of the sign of the binding-energy change will not in every case be accompanied by exact quantitative identities for all other functions. The particular conditions for "trapping" or "improvement" [Eqs. (22) and (24)] are therefore not general. Since each particular mutational change will have consequences which differ from enzyme to enzyme, no expression will satisfy the general case. It can, however, be seen that the effect of increasing complementarity to one substrate will, on the average, result in decreased catalysis with respect to other substrates. This means that negative effects on growth rate will be severest in single-copy products but will tend to be buffered if mutation occurs in a single copy of a multiple array. An increasing range of favourable mutations will become accessible as the copy number increases (see Fig. 4). While the particular functions derived for the simple model will not apply, the general direction of the effect remains.

Finally, we have omitted mention of the effect of multiple copies of the genetic material itself on growth. Apart from the effect on allocation of protein dealt with in the algebra, such an increase will constitute an additional "load" on growth, and this will be the same irrespective of which gene is duplicated. This effect is likely to be relatively small, since each gene copy is acting as a catalyst for the production of many copies of the proto-enzyme. Neglect of this, we believe, does not significantly affect the correctness of our analysis of behaviour of the system. The

same arguments apply to the production of other noncatalytic products (structural proteins) that may be part of the output. Whatever the importance and concentration of such molecules in modern organisms, they would have been a small component of primitive cells.

We therefore feel that these deviations from the simple algebraic model would not significantly affect the general behaviour of the evolving system. "Duplication and divergence" driven by growth-rate increases are seen to be the mechanism for the establishment of the modern metabolic machinery.

References

Cairns-Smith AG (1971) The Life Puzzle. Oliver & Boyd, Edinburgh

Chapman DJ, Ragan MA (1980) Evolution of biochemical pathways: evidence from comparative biochemistry. Annu Rev Plant Physiol 31:639–678

Citri N, Pollock MR (1966) The biochemistry and function of β-lactamase (penicillinase). Adv Enzymol 28:237–323

Dayhoff MO (1976) The origin and evolution of protein superfamilies. Fed Proc 35:3132–3138

Dayhoff MO, Barker WC, Hunt CT, Schwartz RM (1978) Protein superfamilies. In: Dayhoff MO (ed) Atlas of protein sequence and structure, Vol 5, Suppl 3. National Biomedical Research Foundation, Washington DC, pp 9–10

Fersht A (1977) Enzyme Structure and Mechanism. WH Freeman, San Francisco

Flint, HJ, Tateson RW, Barthelmess IB, Porteous DJ, Donachie WD, Kacser H (1981) Control of flux in the arginine pathway of *Neurospora crassa*. Modulations of enzyme activity and concentration. Biochem J 200:231–246

Flory PJ (1967) Configurational statistics of polypeptide chains. In: Ramachandran GN (ed) Conformation of biopolymers, Vol 1. Academic Press, New York, pp 339–363

Goodman M (1981) Globin evolution was apparently very rapid in early vertebrates: a reasonable case against the rate constancy hypothesis. J Mol Evol 17:114–120

Haynes RH, Kunz BA (1981) DNA repair and mutagenesis in yeast. In: Strathern JN, Jones EW, Broach JR (eds) The molecular biology of the yeast *Saccharomyces*: life cycle and inheritance. Cold Spring Harbor Laboratory, Cold Spring Harbor, New York, pp 371–414

Jencks WP (1975) Binding energy, specificity and enzymic catalysis: the Circe effect. Adv Enzymol 43:220–410

Jensen RA (1976) Enzyme recruitment in the evolution of new function. Annu Rev Microbiol 30:409–425

Kacser H, Burns JA (1973) The control of flux. Symp Soc Exp Biol 27:65–104

Kacser H, Burns JA (1979) Molecular democracy: Who shares the controls? Biochem Soc Trans 7:1149–1160

Kacser H, Burns JA (1981) The molecular basis of dominance. Genetics 97:639–666

Maynard Smith J (1961) The limitations of molecular evolution. In: Good IJ (ed) The Scientist Speculates. Heinemann, London, pp 252–256

Ninio J (1982) Molecular Approaches to Evolution. Pitman, London

Privalov PC (1979) Stability of proteins: I. Small globular proteins. Adv Protein Chem 33:167–241

Ptitsyn OB, Finkelstein AV (1980) Similarities of protein topologies: evolutionary divergence, functional convergence or principles of folding? Q Rev Biophys 3:339–386

Remington SJ, Matthews BW (1980) A systematic approach to the comparison of protein structures. J Mol Biol 140:77–99

Richardson JS (1981) The anatomy and taxonomy of protein structure. Adv Protein Chem 34:167–339

Rossmann MG (1981) Evolution of glycolytic enzymes. Philos Trans R Soc Lond [Biol] 293:191–203

Rossmann MG, Argos P (1977) The taxonomy of protein structure. J Mol Biol 109:99–129

Schulz GE (1980) The significance of similarities between protein chain folds. In: Jaenicke R (ed) Protein folding. Elsevier North-Holland Biomedical Press, Amsterdam, pp 199–213

Von Heijne G, Leimar O, Blomberg C (1978) On the emergence of new function in primitive proteins. J Theor Biol 75:167–180

Waley SG (1969) Some aspects of the evolution of metabolic pathways. Comp Biochem Physiol 30:1–11

Witkin EM (1976) Ultraviolet mutagenesis and inducible DNA repair in *E. coli*. Bacteriol Rev 40:869–907

Woese CR (1965) On the evolution of the genetic code. Proc Natl Acad Sci USA 54:1546–1552

Woese CR (1972) The emergence of genetic organisation. In: Ponnamperuma C (ed) Exobiology. North-Holland Publishing Co, Amsterdam and London, p 308

Ycas M (1974) On earlier states of the biochemical system. J Theor Biol 44:145–160

Zuckerkandl E (1975) The appearance of novel structures and functions in proteins during evolution. J Mol Evol 7:1–53

Received June 24, 1983